

Package ‘MplusAutomation’

January 31, 2018

Type Package

Title An R Package for Facilitating Large-Scale Latent Variable Analyses in Mplus

Version 0.7-1

Date 2018-01-31

Maintainer Michael Hallquist <michael.hallquist@gmail.com>

Description Leverages the R language to automate latent variable model estimation and interpretation using 'Mplus', a powerful latent variable modeling program developed by Muthen and Muthen (<<http://www.statmodel.com>>). Specifically, this package provides routines for creating related groups of models, running batches of models, and extracting and tabulating model parameters and fit statistics.

License LGPL-3

URL <https://github.com/michaelhallquist/MplusAutomation>

Depends R (>= 2.15.0), methods

Imports utils, boot, plyr, gsubfn, coda, xtable, lattice, texreg, pander, digest, parallel

Suggests rhdf5, tcltk, relimp, testthat

LazyLoad yes

LazyData yes

RoxygenNote 6.0.1

NeedsCompilation no

Author Michael Hallquist [aut, cre],
Joshua Wiley [aut]

Repository CRAN

Date/Publication 2018-01-31 18:05:22 UTC

R topics documented:

cd	3
coef.mplus.model	4

compareModels	5
confint.mplus.model	7
createModels	9
createSyntax	10
detectVariables	11
extract	12
extractModelParameters	13
extractModelSummaries	16
extractModIndices	19
getSavedata_Bparams	20
getSavedata_Data	21
getSavedata_Fileinfo	22
HTMLSummaryTable	23
LatexSummaryTable	25
lcademo	26
long2LGMM	27
lookupTech1Parameter	30
mplus.traceplot	31
MplusAutomation	32
mplusModeler	33
mplusObject	39
mplusRcov	42
paramExtract	45
parseCatOutput	46
parseMplus	47
plot.mplusObject	48
prepareMplusData	49
print.MplusRstructure	52
readModels	53
runModels	56
runModels_Interactive	57
separateHyphens	58
showSummaryTable	59
summary.mplusObject	60
SummaryTable	61
testBParamCompoundConstraint	63
testBParamConstraint	64
trainLGMM	66
update.mplusObject	69

cd	<i>Change directory</i>
----	-------------------------

Description

The function takes a path and changes the current working directory to the path. If the directory specified in the path does not currently exist, it will be created.

Usage

```
cd(base, pre, num)
```

Arguments

base	a character string with the base path to the directory. This is required.
pre	an optional character string with the prefix to add to the base path. Non character strings will be coerced to character class.
num	an optional character string, prefixed by pre. Non character strings will be coerced to character class.

Details

The function has been designed to be platform independent, although it has had limited testing. Path creation is done using `file.path`, the existence of the directory is checked using `file.exists` and the directory created with `dir.create`. Only the first argument, is required. The other optional arguments are handy when one wants to create many similar directories with a common base.

Value

NULL, changes the current working directory

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

Examples

```
## Not run:
# an example just using the base
cd("~/testdir")

# an example using the optional arguments
base <- "~/testdir"
pre <- "test_"

cd(base, pre, 1)
cd(base, pre, 2)

## End(Not run)
```

coef.mplus.model *Return coefficients for an mplus.model object*

Description

This is a method for returning the coefficients of an mplus.model object. It works directly on an object stored from readModels such as: `object <- readModels("/path/to/model/model.out")`.

Method that calls `coef.mplus.model`. See further documentation there.

Usage

```
## S3 method for class 'mplus.model'
coef(object, type = c("un", "std", "stdy", "stdyx"),
      params = c("regression", "loading", "undirected", "expectation",
                "variability", "new"), ..., raw = FALSE)

## S3 method for class 'mplusObject'
coef(object, ...)
```

Arguments

object	An object of class mplusObject
type	A character vector indicating the type of coefficients to return. One of “un”, “std”, “stdy”, or “stdyx”.
params	A character vector indicating what type of parameters to extract. Any combination of “regression”, “loading”, “undirected”, “expectation”, “variability”, and “new”. A single one can be passed or multiple. By default, all are used and all parameters are returned.
...	Additional arguments to pass on (not currently used)
raw	A logical defaulting to FALSE indicating whether to parse and return coefficients based on the type (regression, etc.) and relabel using an arrow notation, or to return the raw coefficients in a named vector.

Value

Either a data frame of class ‘mplus.model.coefs’, or in the case of multiple group models, a list of class ‘mplus.model.coefs’, where each element of the list is a data frame of class ‘mplus.model.coefs’, or a named vector of coefficients, if `raw=TRUE`.

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

See Also

[readModels](#)

Other Mplus-Formatting: [confint.mplus.model](#), [extract](#), [print.MplusRstructure](#), [summary.mplusObject](#)

Examples

```
## Not run:
# simple example of a model using builtin data
# demonstrates use
test <- mplusObject(
  TITLE = "test the MplusAutomation Package;",
  MODEL = "
    mpg ON wt hp;
    wt WITH hp;";
  OUTPUT = "STANDARDIZED;";
  usevariables = c("mpg", "wt", "hp"),
  rdata = mtcars)

res <- mplusModeler(test, "mtcars.dat", modelout = "model1.inp", run = 1L)

# example of the coef method on an mplus.model object
# note that res$results holds the results of readModels()
coef(res$results)
coef(res$results, type = "std")
coef(res$results, type = "stdy")
coef(res$results, type = "stdyx")

# there is also a method for mplusObject class
coef(res)

# remove files
unlink("mtcars.dat")
unlink("model1.inp")
unlink("model1.out")
unlink("Mplus Run Models.log")

## End(Not run)
```

compareModels

Compare the output of two Mplus models

Description

The `compareModels` function compares the output of two Mplus files and prints similarities and differences in the model summary statistics and parameter estimates. Options are provided for filtering out fixed parameters and nonsignificant parameters. When requested, `compareModels` will compute the chi-square difference test for nested models (does not apply to MLMV, WLSM, and WLSMV estimators, where DIFFTEST in Mplus is needed). Model outputs to be compared can be full summaries and parameters (generated by `readModels`), summary statistics only (`extractModelSummaries`), or parameters only (`extractModelParameters`).

Usage

```
compareModels(m1, m2, show = "all", equalityMargin = c(param = 1e-04, pvalue
```

```
= 1e-04), compare = "unstandardized", sort = "none", showFixed = FALSE,
showNS = TRUE, diffTest = FALSE)
```

Arguments

m1	The first Mplus model to be compared. Generated by readModels, extractModelSummaries, or extractModelParameters.
m2	The second Mplus model to be compared.
show	What aspects of the models should be compared. Options are "all", "summaries", "equal", "diff", "pdiff", and "unique". See below for details.
equalityMargin	Defines the discrepancy between models that is considered equal. Different margins can be specified for p-value equality versus parameter equality. Defaults to .0001 for both.
compare	Which parameter estimates should be compared. Options are "unstandardized", "stdyx.standardized" "stdy.standardized", and "std.standardized".
sort	How to sort the output of parameter comparisons. Options are "none", "type", "alphabetical", and "maxDiff". See below for details.
showFixed	Whether to display fixed parameters in the output (identified where the est/se = 999.000, per Mplus convention). Default to FALSE.
showNS	Whether to display non-significant parameter estimates. Can be TRUE or FALSE, or a numeric value (e.g., .10) that defines what p-value is filtered as non-significant.
diffTest	Whether to compute a chi-square difference test between the models. Assumes that the models are nested. Not available for MLMV, WLSMV, and ULSMV estimators. Use DIFFTEST in Mplus instead.

Details

The show parameter can be one or more of the following, which can be passed as a vector, such as c("equal", "pdiff").

show "all" Display all available model comparison. Equivalent to c("summaries", "equal", "diff", "pdiff", "unique").

"summaries" Print a comparison of model summary statistics. Compares the following summary statistics (where available): c("Title", "Observations", "Estimator", "Parameters", "LL", "AIC", "BIC", "ChiSqM_Value", "ChiSqM_DF", "CFI", "TLI", "RMSEA", "SRMR", "WRMR")

"allsummaries" Prints a comparison of all summary statistics available in each model. May generate a lot of output.

"equal" Print parameter estimates that are equal between models (i.e., \leq equalityMargin["param"]).

"diff" Print parameter estimates that are different between models (i.e., $>$ equalityMargin["param"]).

"pdiff" Print parameter estimates where the p-values differ between models (i.e., $>$ equalityMargin["pvalue"]).

"unique" Print parameter estimates that are unique to each model.

The sort parameter determines the order in which parameter estimates are displayed. The following options are available:

sort "none" No sorting is performed, so parameters are output in the order presented in Mplus. (Default)

"type" Sort parameters by their role in the model. This groups output by regression coefficient (ON), factor loadings (BY), covariances (WITH), and so on. Within each type, output is alphabetical.

"alphabetical" Sort parameters in alphabetical order.

"maxDiff" Sort parameter output by the largest differences between models (high to low).

Value

No value is returned by this function. It is used to print model differences to the R console.

Author(s)

Michael Hallquist

Examples

```
# make me!!!
```

`confint.mplus.model` *Return confidence intervals for an mplus.model object*

Description

This is a method for returning the confidence of an `mplus.model` object. It works directly on an object stored from `readModels` such as: `object <- readModels("/path/to/model/model.out")`.

Method that calls `confint.mplus.model`. See further documentation there.

Usage

```
## S3 method for class 'mplus.model'
confint(object, parm, level = 0.95, type = c("un",
  "std", "stdy", "stdyx"), params = c("regression", "loading", "undirected",
  "expectation", "variability", "new"), ...)
```

```
## S3 method for class 'mplusObject'
confint(object, ...)
```

Arguments

<code>object</code>	An object of class <code>mplusObject</code>
<code>parm</code>	Included as all <code>confint()</code> methods must include it. Not used currently for Mplus.
<code>level</code>	A numeric vector indicating the level of confidence interval to extract. Options are .95, .90, or .99 as those are all Mplus provides.

type	A character vector indicating the type of confidence intervals to return. One of “un”, “std”, “stdy”, or “stdyx”.
params	A character vector indicating what type of parameters to extract. Any combination of “regression”, “loading”, “undirected”, “expectation”, “variability”, and “new”. A single one can be passed or multiple. By default, all are used and all parameters are returned.
...	Additional arguments to pass on (not currently used)

Value

A data frame of class ‘mplus.model.cis’, or in the case of multiple group models, a list of class ‘mplus.model.cis’, where each element of the list is a data frame of class ‘mplus.model.cis’.

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

See Also

[readModels](#)

Other Mplus-Formatting: [coef.mplus.model](#), [extract](#), [print.MplusRstructure](#), [summary.mplusObject](#)

Examples

```
## Not run:
# simple example of a model using builtin data
# demonstrates use
test <- mplusObject(
  TITLE = "test the MplusAutomation Package;",
  MODEL = "
    mpg ON wt hp;
    wt WITH hp;"
  OUTPUT = "STANDARDIZED; CINTERVAL;"
  usevariables = c("mpg", "wt", "hp"),
  rdata = mtcars)

res <- mplusModeler(test, "mtcars.dat", modelout = "model1.inp", run = 1L)

# example of the confint method on an mplus.model object
# note that res$results holds the results of readModels()
confint(res$results)
confint(res$results, type = "std")
confint(res$results, type = "stdy")
confint(res$results, type = "stdyx", level = .99)

# there is also a method for mplusObject class
confint(res)
screenreg(res, cis = TRUE, single.row = TRUE)

# remove files
unlink("mtcars.dat")
```



```
unlink("model1.inp")
unlink("model1.out")
unlink("Mplus Run Models.log")

## End(Not run)
```

createModels

Create Mplus Input Files from Template

Description

The createModels function processes a single Mplus template file and creates a group of related model input files. Definitions and examples for the template language are provided in the MplusAutomation vignette and are not duplicated here at the moment. See this documentation: vignette("Vignette", package="MplusAutomation")

Usage

```
createModels(templatefile)
```

Arguments

templatefile The filename (absolute or relative path) of an Mplus template file to be processed. Example "C:/MplusTemplate.txt"

Value

No value is returned by this function. It is solely used to process an Mplus template file.

Author(s)

Michael Hallquist

Examples

```
## Not run:
  createModels("L2 Multimodel Template No iter.txt")

## End(Not run)
```

`createSyntax`*Create the Mplus input text for an mplusObject*

Description

This function takes an object of class `mplusObject` and creates the Mplus input text corresponding to it, including data link and variable names.

Usage

```
createSyntax(object, filename, check = TRUE, add = FALSE, imputed = FALSE)
```

Arguments

<code>object</code>	An object of class <code>mplusObject</code>
<code>filename</code>	The name of the data file as a character vector
<code>check</code>	A logical indicating whether or not to run <code>parseMplus</code> on the created input file. Checks for errors like lines that are too long, or for missing semi-colons and gives notes.
<code>add</code>	A logical passed on to <code>parseMplus</code> whether to add semi colons to line ends. Defaults to FALSE.
<code>imputed</code>	A logical whether the data are multiply imputed. Defaults to FALSE.

Value

A character string containing all the text for the Mplus input file.

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

See Also

[prepareMplusData](#), [mplusModeler](#)

Examples

```
# example mplusObject
example1 <- mplusObject(MODEL = "mpg ON wt;",
  usevariables = c("mpg", "hp"), rdata = mtcars)

# create the Mplus input text
cat(createSyntax(example1, "example1.dat"), fill=TRUE)

# update the object, then create input text
cat(createSyntax(update(example1,
  TITLE = ~ "This is my title;"),
```

```
MODEL = ~ . + "\nmpg ON hp;",  
usevariables = c("mpg", "hp", "wt"), "example1.dat"),  
fill=TRUE)  
rm(example1)
```

detectVariables	<i>Automatically detect variables from an Mplus model object</i>
-----------------	--

Description

This is a function to automatically detect the variables used in an Mplus model object.

Usage

```
detectVariables(object)
```

Arguments

object An Mplus model object from `mplusObject.#`'

Value

A vector of variables from the R dataset to use.

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

See Also

[mplusModeler](#), [mplusObject](#)

Examples

```
example1 <- mplusObject(MODEL = "mpg ON wt;",  
  rdata = mtcars, autov = FALSE)  
example1$usevariables  
MplusAutomation::detectVariables(example1)  
  
example2 <- mplusObject(MODEL = "mpg ON wt;",  
  rdata = mtcars, autov = TRUE)  
example2$usevariables  
example3 <- update(example2,  
  MODEL = ~ . + "mpg ON qsec; wt WITH qsec;",  
  autov = TRUE)  
example3$usevariables  
rm(example1, example2, example3)
```

 extract

*Extract function to make Mplus output work with the **texreg** package*

Description

This is a method for extracting output in a format suitable for the **texreg** package. Uses `coef` for most the work.

Usage

```
extract.mplus.model(model, summaries = "none", cis = FALSE,
  escape.latex = FALSE, ...)

extract.mplusObject(model, summaries = "none", cis = FALSE, ...)

## S4 method for signature 'mplus.model'
extract(model, summaries = "none", cis = FALSE,
  escape.latex = FALSE, ...)

## S4 method for signature 'mplusObject'
extract(model, summaries = "none", cis = FALSE, ...)
```

Arguments

<code>model</code>	An Mplus model object. This typically comes either from readModels directly, or indirectly via mplusModeler . The results will have different classes, but <code>extract</code> methods are defined for both.
<code>summaries</code>	A character vector which summaries to include. Defaults to “none”.
<code>cis</code>	A logical whether to extract confidence intervals.
<code>escape.latex</code>	A logical value whether to escape dollar signs in coefficient names for LaTeX. Defaults to FALSE.
<code>...</code>	Additional arguments passed to coef.mplus.model .

Value

A `texreg` object, or for multiple group models, a list of `texreg` objects.

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

See Also

[readModels](#)

Other Mplus-Formatting: [coef.mplus.model](#), [confint.mplus.model](#), [print.MplusRstructure](#), [summary.mplusObject](#)

Examples

```

## Not run:
# simple example of a model using builtin data
# demonstrates use
test <- mplusObject(
  TITLE = "test the MplusAutomation Package;",
  MODEL = "
    mpg ON wt hp;
    wt WITH hp;";
  OUTPUT = "STANDARDIZED;";
  usevariables = c("mpg", "wt", "hp"),
  rdata = mtcars)

res <- mplusModeler(test, "mtcars.dat", modelout = "model1.inp", run = 1L)

extract(res$results)
# there is also a method for mplusObject class
extract(res)

# load the texreg package
# to use pretty printing via screenreg
# uncomment to run these examples
# library(texreg)
# screenreg(res)
# screenreg(res, type = 'stdyx')

# screenreg(res, type = 'un', params = 'regression',
#   single.row=TRUE)
# screenreg(res, type = 'un', params = 'regression', summaries = 'CFI',
#   single.row=TRUE)

# remove files
unlink("mtcars.dat")
unlink("model1.inp")
unlink("model1.out")
unlink("Mplus Run Models.log")

## End(Not run)

```

```
extractModelParameters
```

Extract model parameters from MODEL RESULTS section.

Description

Extracts the model parameters from the MODEL RESULTS section of one or more Mplus output files. If a particular output file has more than one results section (unstandardized, stdyx, stdy, and/or std), a list will be returned. If the target is a directory, all .out files therein will be parsed and a single list will be returned, where the list elements are named by the output file name. Returned parameters often include the parameter estimate, std. err, param/s.e., and two-tailed p-value.

Usage

```
extractModelParameters(target = getwd(), recursive = FALSE, filefilter,
  dropDimensions = FALSE, resultType)
```

Arguments

target	the directory containing Mplus output files (.out) to parse OR the single output file to be parsed. May be a full path, relative path, or a filename within the working directory. Defaults to the current working directory. Example: "C:/Users/Michael/Mplus Runs"
recursive	optional. If TRUE, parse all models nested in subdirectories within target. Defaults to FALSE.
filefilter	a Perl regular expression (PCRE-compatible) specifying particular output files to be parsed within directory. See regex or http://www.pcre.org/pcre.txt for details about regular expression syntax.
dropDimensions	Relevant only for multi-file parsing. If TRUE, then if only one output section (usually unstandardized) is present for all files in the parsed list, then eliminate the second-level list (which contains elements for each output section). The result is that the elements of the returned list are data.frame objects with the relevant parameters.
resultType	N.B.: this parameter is deprecated and will be removed in a future version. The new default is to extract all results that are present and return a list (see below for details). resultType specified the results section to extract. If raw, the unstandardized estimates will be returned. "stdyx", "stdy", and "std" are the other options, which extract different standardized solutions. See the Mplus User's Guide for additional details about the differences in these standardizations.

Value

If target is a single file, a list containing unstandardized and standardized results will be returned. If all standardized solutions are available, the list element will be named: unstandardized, stdyx.standardized, stdy.standardized, and std.standardized. If confidence intervals are output using OUTPUT:CINTERVAL, then a list element named ci.unstandardized will be included. Each of these list elements is a data.frame containing relevant model parameters.

If target is a directory, a list will be returned, where each element contains the results for a single file, and the top-level elements are named after the corresponding output file name. Each element within this list is itself a list, with elements as in the single file case above.

The core data.frame for each MODEL RESULTS section typically has the following structure:

paramHeader	The header that begins a given parameter set. Example: "FACTOR1 BY"
param	The particular parameter being measured (within paramHeader). Example: "ITEM1"
est	Parameter estimate value.
se	Standard error of the estimate
est_se	Quotient of est/se, representing z-test/t-test in large samples
pval	Two-tailed p-value for the est_se quotient.

In the case of output from Bayesian estimation (ESTIMATOR=BAYES), the `data.frame` will contain a different set of variables, including some of the above, as well as

<code>posterior_sd</code>	Posterior standard deviation of the estimate.
<code>lower_2.5ci</code>	Lower 2.5 percentile of the estimate.
<code>upper_2.5ci</code>	Upper 2.5 percentile (aka 97.5 percentile) of the estimate.

Also note that the `pval` column for Bayesian output represents a one-tailed estimate.

In the case of output from a Monte Carlo study (MONTECARLO: and MODEL POPULATION:), the `data.frame` will contain a different set of variables, including some of the above, as well as

<code>population</code>	Population parameter value.
<code>average</code>	Average parameter estimate across replications.
<code>population_sd</code>	Standard deviation of parameter value in population across replications.
<code>average_se</code>	Average standard error of estimated parameter value across replications.
<code>mse</code>	Mean squared error.
<code>cover_95</code>	Proportion of replications whose 95% confidence interval for the parameter includes the population value.
<code>pct_sig_coef</code>	Proportion of replications for which the two-tailed significance test of the parameter is significant ($p < .05$).

In the case of confidence interval output (OUTPUT:CINTERVAL), the list element `ci.unstandardized` will contain a different set of variables, including some of the above, as well as

<code>low.5</code>	Lower 0.5% CI estimate.
<code>low2.5</code>	Lower 2.5% CI estimate.
<code>low5</code>	Lower 5% CI estimate.
<code>est</code>	Parameter estimate value.
<code>up5</code>	Upper 5% (i.e., 95%) CI estimate.
<code>up2.5</code>	Upper 2.5% (i.e., 97.5%) CI estimate.
<code>up.5</code>	Upper 0.5% (i.e., 99.5%) CI estimate.

If the model contains multiple latent classes, an additional variable, `LatentClass`, will be included, specifying the latent class number. Also, the Categorical Latent Variables section will be included as `LatentClass "Categorical.Latent.Variables."`

If the model contains multiple groups, `Group` will be included.

If the model contains two-level output (between/within), `BetweenWithin` will be included.

Author(s)

Michael Hallquist

See Also

[extractModelSummaries](#)

Examples

```
## Not run:
ex3.14 <- extractModelParameters(
"C:/Program Files/Mplus/Mplus Examples/User's Guide Examples/ex3.14.out")

## End(Not run)
```

extractModelSummaries *Extract summary statistics from a single output file or from a group of Mplus models within a directory*

Description

Parses a group of Mplus model output files (.out extension) for model fit statistics. At this time, the details extracted are fixed and include: Filename, InputInstructions, Title, Estimator, LL, BIC, aBIC, AIC, AICC. The infrastructure is in place to allow for user-specified selection of summary statistics in future versions.

Usage

```
extractModelSummaries(target = getwd(), recursive = FALSE, filefilter)
```

Arguments

target	the directory containing Mplus output files (.out) to parse OR the single output file to be parsed. Defaults to the current working directory. Example: "C:/Users/Michael/Mplus Runs"
recursive	optional. If TRUE, parse all models nested in subdirectories within directory. Defaults to FALSE.
filefilter	a Perl regular expression (PCRE-compatible) specifying particular output files to be parsed within directory. See regex or http://www.pcre.org/pcre.txt for details about regular expression syntax.

Value

Returns a data.frame containing model fit statistics for all output files within directory. The data.frame contains some of the following variables (depends on model type):

Title	Title for the model, specified by the TITLE: command
Filename	Filename of the output file
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)
DIC	Deviance Information Criterion. Available in ESTIMATOR=BAYES output.
Parameters	Number of parameters estimated by the model
pD	Estimated number of parameters in Bayesian output
Observations	The number of observations for the model (does not support 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
ChiSqM_ScalingCorrection	H0 Scaling Correction Factor
ObsRepChiSqDiff_95CI_LB	Lower bound of 95% confidence interval for the difference between observed and replicated chi-square values
ObsRepChiSqDiff_95CI_UB	Upper bound of 95% confidence interval for the difference between observed and replicated chi-square values
PostPred_PValue	Posterior predictive p-value
PriorPostPred_PValue	Prior Posterior Predictive P-Value
BLRT_RequestedDraws	Number of requested bootstrap draws for TECH14.
BLRT_KM1LL	Log-likelihood of the K-1 model (one less class) for the Bootstrapped Likelihood Ratio Test (TECH14).
BLRT_2xLLDiff	Two times the log-likelihood difference of the models with K and K-1 classes (TECH14).
BLRT_ParamDiff	Difference in the number of parameters for models with K and K-1 classes (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_SuccessfulDraws	The number of successful bootstrapped samples used in the Bootstrapped Likelihood Ratio Test
SRMR	Standardized root mean square residual

SRMR.Between	For TYPE=TWOLEVEL output, standardized root mean square residual for between level
SRMR.Within	For TYPE=TWOLEVEL output, standardized root mean square residual for within level
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
NumFactors	For TYPE=EFA output, the number of factors
T11_KM1Starts	TECH11: Number of initial stage random starts for k-1 model
T11_KM1Final	TECH11: Number of final stage optimizations for k-1 model
T11_KM1LL	TECH11: Log-likelihood of the K-1 model used for the Vuong-Lo-Mendell-Rubin LRT
T11_VLMR_2xLLDiff	TECH11: 2 * Log-likelihood Difference of K-class vs. K-1-class model for the Vuong-Lo-Mendell-Rubin LRT
T11_VLMR_ParamDiff	TECH11: Difference in number of parameters between K-class and K-1-class model for the Vuong-Lo-Mendell-Rubin LRT
T11_VLMR_Mean	TECH11: Vuong-Lo-Mendell-Rubin LRT mean
T11_VLMR_SD	TECH11: Vuong-Lo-Mendell-Rubin LRT standard deviation
T11_VLMR_PValue	TECH11: Vuong-Lo-Mendell-Rubin LRT p-value
T11_LMR_Value	TECH11: Lo-Mendell-Rubin Adjusted LRT value
T11_LMR_PValue	TECH11: Lo-Mendell-Rubin Adjusted LRT p-value

Author(s)

Michael Hallquist

See Also

[regex](#), [runModels](#), [readModels](#)

Examples

```
## Not run:
allExamples <- extractModelSummaries(
  "C:/Program Files/Mplus/Mplus Examples/User's Guide Examples")

## End(Not run)
```

extractModIndices *Extract model modification indices.*

Description

Extracts the model modification indices from the MODEL MODIFICATION INDICES section of one or more Mplus output files. If the `target` is a directory, all `.out` files therein will be parsed and a single list will be returned, where the list elements are named by the output file name. Returned parameters typically include the pairwise relationships between variables to be freed, the change in model chi-square (M.I.), and the expected parameter change (E.P.C.).

Usage

```
extractModIndices(target = getwd(), recursive = FALSE, filefilter)
```

Arguments

<code>target</code>	the directory containing Mplus output files (<code>.out</code>) to parse OR the single output file to be parsed. May be a full path, relative path, or a filename within the working directory. Defaults to the current working directory. Example: “C:/Users/Michael/Mplus Runs”
<code>recursive</code>	optional. If TRUE, parse all models nested in subdirectories within <code>target</code> . Defaults to FALSE.
<code>filefilter</code>	a Perl regular expression (PCRE-compatible) specifying particular output files to be parsed within directory. See <code>regex</code> or http://www.pcre.org/pcre.txt for details about regular expression syntax.

Value

If `target` is a single file, a `data.frame` containing modification results for the target output file will be returned. If `target` is a directory, a list will be returned, where each element contains a `data.frame` of the modification indices for a single file, and the top-level elements are named after the corresponding output file name. The basic `data.frame` containing the MODEL MODIFICATION INDICES section of `outfile`. Variables include

<code>modV1</code>	The first variable in the pair to be freed according to the M.I.
<code>operator</code>	The suggested relationship between <code>modV1</code> and <code>modV2</code> (e.g., WITH for freeing the covariance between <code>modV1</code> and <code>modV2</code>)
<code>modV2</code>	The first variable in the pair to be freed according to the M.I.
<code>MI</code>	The decrease in model chi-square if the specified relationship is freely estimated
<code>EPC</code>	The expected parameter estimate between <code>modV1</code> and <code>modV2</code> if freed.
<code>Std_EPC</code>	The EPC value standardized using the variances of the continuous latent variables.
<code>StdYX_EPC</code>	The EPC value standardized using the variances of the continuous latent variables as well as the variances of the background and/or outcome variables.

Author(s)

Michael Hallquist

See Also[readModels](#), [extractModelSummaries](#), [extractModelParameters](#)**Examples**

```
## Not run:
ex3.14 <- extractModIndices(
"C:/Program Files/Mplus/Mplus Examples/User's Guide Examples/ex3.14.out")

## End(Not run)
```

getSavedata_Bparams	<i>Load the draws from the Bayesian model posterior distribution (SAVE-DATA BPARAMETERS) command into an R data.frame</i>
---------------------	---

Description

This function reads a the BPARAMETERS output file from the Mplus SAVEDATA BPARAMETERS command and returns an R data.frame object.

Usage

```
getSavedata_Bparams(outfile, discardBurnin = TRUE)
```

Arguments

outfile	Required. The name of the Mplus output file to read. Can be an absolute or relative path. If outfile is a relative path or just the filename, then it is assumed that the file resides in the working directory getwd().
discardBurnin	Optional. Whether to discard the burn-in phase of each MCMC chain (i.e., the first half).

Value

A list containing the draws from the MCMC chains for a Bayesian model that uses the SAVE-DATA BPARAMETERS command. Each list element corresponds to a single MCMC chain, as specified by the ANALYSIS: CHAINS syntax in Mplus. If discardBurnin is FALSE, then a superordinate list is provided that divides output in terms of burn-in versus valid draw halves of the MCMC chains. For documentation of how Mplus implements chain convergence checks and MCMC draws, see here: <http://www.statmodel.com/download/Bayes3.pdf>.

Note

Note that the `outfile` parameter should refer to the Mplus output file (.out extension), not the actual dataset generated by `SAVEDATA`. This function reads information about the dataset from the .out file and loads the dataset accordingly.

Author(s)

Michael Hallquist, Florian Boeing-Messing

References

<http://www.statmodel.com/download/Bayes3.pdf>

See Also

[getSavedata_Fileinfo](#), [getSavedata_Data](#)

Examples

```
## Not run:  
fileInfo <- getSavedata_Data("C:/Program Files/Mplus/Test Output.out")  
  
## End(Not run)
```

<code>getSavedata_Data</code>	<i>Load an analysis dataset from the <code>SAVEDATA</code> command into an R <code>data.frame</code></i>
-------------------------------	--

Description

This function reads an analysis dataset generated by the Mplus `SAVEDATA` command and returns an R `data.frame` object.

Usage

```
getSavedata_Data(outfile)
```

Arguments

<code>outfile</code>	Required. The name of the Mplus output file to read. Can be an absolute or relative path. If <code>outfile</code> is a relative path or just the filename, then it is assumed that the file resides in the working directory <code>getwd()</code> .
----------------------	---

Value

A `data.frame` containing the analysis dataset generated by the `SAVEDATA` command.

Note

Note that the `outfile` parameter should refer to the Mplus output file (.out extension), not the actual dataset generated by `SAVEDATA`. This function reads information about the dataset from the .out file and loads the dataset accordingly.

Author(s)

Michael Hallquist

See Also

[getSavedata_Fileinfo](#)

Examples

```
## Not run:
  savedat <- getSavedata_Data("C:/Program Files/Mplus/Test Output.out")

## End(Not run)
```

`getSavedata_Fileinfo` *Read Variable Names, Formats, and Widths from data generated by the SAVEDATA Command*

Description

This function reads the `SAVEDATA INFORMATION` section from an Mplus output file that used the `SAVEDATA` command, and it returns a list with the filename, variable names, variable formats, and variable widths of the `SAVEDATA` file. If present, the function also parses information about the Bayesian Parameters (`BPARAMETERS`) file.

Usage

```
getSavedata_Fileinfo(outfile)
```

Arguments

<code>outfile</code>	required. The name of the Mplus output file to read. Can be an absolute or relative path. If <code>outfile</code> is a relative path or just the filename, then it is assumed that the file resides in the working directory <code>getwd()</code> .
----------------------	---

Value

Returns a list of `SAVEDATA` file information that includes:

<code>fileName</code>	The name of the file containing the analysis dataset created by the Mplus <code>SAVEDATA</code> command.
<code>fileVarNames</code>	A character vector containing the names of variables in the dataset.

fileVarFormats	A character vector containing the Fortran-style formats of variables in the dataset.
fileVarWidths	A numeric vector containing the widths of variables in the dataset (which is stored in fixed-width format).
bayesFile	The name of the BPARAMETERS file containing draws from the posterior distribution created by the Mplus SAVEDATA BPARAMETERS command.
bayesVarNames	A character vector containing the names of variables in the BPARAMETERS dataset.
tech3File	A character vector of the tech 3 output.
tech4File	A character vector of the tech 4 output.

Author(s)

Michael Hallquist

See Also

[getSavedata_Data](#)

Examples

```
## Not run:  
fileInfo <- getSavedata_Fileinfo("C:/Program Files/Mplus/Test Output.out")  
  
## End(Not run)
```

HTMLSummaryTable	<i>Create an HTML file containing a summary table of Mplus model statistics</i>
------------------	---

Description

Creates an HTML file containing a summary table of model fit statistics extracted using the `extractModelSummaries` function. By default, the following summary statistics are included: Title, LL, Parameters, AIC, AICC, BIC, RMSEA_Es but these are customizable using the `keepCols` and `dropCols` parameters.

Usage

```
HTMLSummaryTable(modelList, filename = file.path(getwd(),  
"Model Comparison.html"), keepCols, dropCols, sortBy, display = FALSE)
```

Arguments

<code>modellist</code>	A list of models (as a <code>data.frame</code>) returned from the <code>extractModelSummaries</code> function.
<code>filename</code>	The name of the HTML file to be created. Can be an absolute or relative path. If <code>filename</code> is a relative path or just the filename, then it is assumed that the file resides in the working directory <code>getwd()</code> . Example: <code>"Mplus Summary.html"</code>
<code>keepCols</code>	A vector of character strings indicating which columns/variables to display in the summary. Only columns included in this list will be displayed (all others excluded). By default, <code>keepCols</code> is: <code>c("Title", "LL", "Parameters", "AIC", "AICC", "BIC", "RMSE")</code> . Example: <code>c("Title", "LL", "AIC", "CFI")</code>
<code>dropCols</code>	A vector of character strings indicating which columns/variables to omit from the summary. Any column not included in this list will be displayed. By default, <code>dropCols</code> is <code>NULL</code> . Example: <code>c("InputInstructions", "TLI")</code>
<code>sortBy</code>	optional. Field name (as character string) by which to sort the table. Typically an information criterion (e.g., "AIC" or "BIC") is used to sort the table. Defaults to "AICC".
<code>display</code>	optional. This parameter specifies whether to display the table in a web browser upon creation (TRUE or FALSE).

Value

No value is returned by this function. It is solely used to create an HTML file containing summary statistics.

Note

You must choose between `keepCols` and `dropCols` because it is not sensible to use these together to include and exclude columns. The function will error if you include both parameters.

Author(s)

Michael Hallquist

See Also

[extractModelSummaries](#), [showSummaryTable](#), [LatexSummaryTable](#)

Examples

```
# make me!!!
```

LatexSummaryTable	<i>Display summary table of Mplus model statistics in separate window</i>
-------------------	---

Description

Creates a LaTeX-formatted summary table of model fit statistics extracted using the `extractModelSummaries` function. The table syntax is returned by the function, which is useful for embedding LaTeX tables using Sweave. By default, the following summary statistics are included: Title, LL, Parameters, AIC, AICC, BIC, RMSE but these are customizable using the `keepCols` and `dropCols` parameters.

Usage

```
LatexSummaryTable(modellist, keepCols, dropCols, sortBy, label = NULL,
  caption = NULL)
```

Arguments

<code>modellist</code>	A list of models (as a <code>data.frame</code>) returned from the <code>extractModelSummaries</code> function.
<code>keepCols</code>	A vector of character strings indicating which columns/variables to display in the summary. Only columns included in this list will be displayed (all others excluded). By default, <code>keepCols</code> is: <code>c("Title", "LL", "Parameters", "AIC", "AICC", "BIC", "RMSE")</code> Example: <code>c("Title", "LL", "AIC", "CFI")</code>
<code>dropCols</code>	A vector of character strings indicating which columns/variables to omit from the summary. Any column not included in this list will be displayed. By default, <code>dropCols</code> is <code>NULL</code> . Example: <code>c("InputInstructions", "TLI")</code>
<code>sortBy</code>	optional. Field name (as character string) by which to sort the table. Typically an information criterion (e.g., "AIC" or "BIC") is used to sort the table. Defaults to "AICC"
<code>label</code>	optional. A character string specifying the label for the LaTeX table, which can be used for referencing the table.
<code>caption</code>	optional. A character string specifying the caption for the LaTeX table.

Value

A LaTeX-formatted table summarizing the `modellist` is returned (created by `xtable`).

Note

You must choose between `keepCols` and `dropCols` because it is not sensible to use these together to include and exclude columns. The function will error if you include both parameters.

Author(s)

Michael Hallquist

See Also

[extractModelSummaries](#), [HTMLSummaryTable](#), [showSummaryTable](#), [Sweave](#)

Examples

```
# make me!!!
```

lcademo

Latent Class Analysis Demonstration

Description

A list containing the Mplus Data, Output Files, and GH5 Files for a demonstration of using MplusAutomation for latent class analysis. Generated by the vignette on latent class analysis.

Usage

```
lcademo
```

Format

A list containing 11 elements.

Data 2 Class LCA data simulated using Mplus

CFA Mplus output file for CFA

LCA2 Mplus output file for 2 class LCA

LCA3 Mplus output file for 3 class LCA

LCA4 Mplus output file for 4 class LCA

LCA5 Mplus output file for 5 class LCA

CFAGH5 GH5 file for CFA

LCA2GH5 GH5 file for 2 class LCA

LCA3GH5 GH5 file for 3 class LCA

LCA4GH5 GH5 file for 4 class LCA

LCA5GH5 GH5 file for 5 class LCA

long2LGMM

*Long data to wide latent growth mixture model***Description**

This function streamlines the process of converting long data into a format that Mplus can use for latent growth mixture models in wide form. It makes use of continuous time scores, and these time scores must be supplied as variables in the R dataset. For the conversion to wide form, it is assumed that although assessments may have happened in continuous time, a discrete number of assessments (likely isimilar for all participants) were collected.

Usage

```
long2LGMM(data, idvar, assessmentvar, dv, timevars, misstrick = TRUE,
           k = 1L, title = "Trajectory Model", base = "trajmodel_", run = FALSE,
           processors = 1L, starts = "500 100", newdata, cov = c("un",
           "independent", "intercept", "zero"), model)
```

Arguments

data	A data frame in long format (i.e., multiple rows per ID).
idvar	A character string of the variable name in the dataset that is the ID variable.
assessmentvar	A character string of the variable name in the dataset that indicates the particular assessment point for each timepoint.
dv	A character string of the dependent variable name.
timevars	A character vector of the time variables. Can be a single variable or more than one. By allowing more than one variable, it is easy to include linear; linear and quadratic; it is also possible to calculate splines in R and pass these. The variable names should be 7 characters or fewer, each.
misstrick	A logical value whether to set values of the DV where a time variable is missing to missing as well. Defaults to TRUE.
k	An integer indicating the number of distinct classes to test. Currently must be greater than 0 and less than 10.
title	A character string giving a title for the model
base	A character string providing a base name for model outputs, that is combined with the number of classes.
run	A logical value whether or not to run the models or only create the data and input files, but not run them.
processors	An integer value indicating the number of processors to use.
starts	A character string passed to Mplus providing the number of random starts and iterations
newdata	A data frame of new values to use for generating predicted trajectories by class.
cov	A character string indicating the random covariance structure to use
model	An optional argument, can pass an existing model, the output from <code>mplusModeler()</code> .

Details

One valuable feature of this function is that it makes it possible to feed any continuous time scores to Mplus for mixture modelling. For example, continuous linear time is straightforward, but so are quadratic time models or piecewise models. Using facilities in R, spline models are also comparatively easy to specify.

Examples

```
## Not run:
## Simulate Some Data from 3 classes
library(MASS)
set.seed(1234)
allcoef <- rbind(
  cbind(1, mvrnorm(n = 200,
                  mu = c(0, 2, 0),
                  Sigma = diag(c(.2, .1, .01)),
                  empirical = TRUE)),
  cbind(2, mvrnorm(n = 200,
                  mu = c(-3.35, 2, 2),
                  Sigma = diag(c(.2, .1, .1)),
                  empirical = TRUE)),
  cbind(3, mvrnorm(n = 200,
                  mu = c(3.35, 2, -2),
                  Sigma = diag(c(.2, .1, .1)),
                  empirical = TRUE)))
allcoef <- as.data.frame(allcoef)
names(allcoef) <- c("Class", "I", "L", "Q")
allcoef$ID <- 1:nrow(allcoef)
d <- do.call(rbind, lapply(1:nrow(allcoef), function(i) {
  out <- data.frame(
    ID = allcoef$ID[i],
    Class = allcoef$Class[i],
    Assess = 1:11,
    x = sort(runif(n = 11, min = -2, max = 2)))
  out$y <- rnorm(11,
    mean = allcoef$I[i] + allcoef$L[i] * out$x + allcoef$Q[i] * out$x^2,
    sd = .1)
  return(out)
}))

## create splines
library(splines)
time_splines <- ns(d$x, df = 3, Boundary.knots = quantile(d$x, probs = c(.02, .98)))
d$t1 <- time_splines[, 1]
d$t2 <- time_splines[, 2]
d$t3 <- time_splines[, 3]
d$xq <- d$x^2

## create new data to be used for predictions
nd <- data.frame(ID = 1,
  x = seq(from = -2, to = 2, by = .1))
nd.splines <- with(attributes(time_splines),
```

```

        ns(nd$x, df = degree, knots = knots,
           Boundary.knots = Boundary.knots))
nd$t1 <- nd.splines[, 1]
nd$t2 <- nd.splines[, 2]
nd$t3 <- nd.splines[, 3]
nd$xq <- nd$x^2

## create a tuning grid of models to try
## all possible combinations are created of different time trends
## different covariance structures of the random effects
## and different number of classes
tuneGrid <- expand.grid(
  dv = "y",
  timevars = list(c("t1", "t2", "t3"), "x", c("x", "xq")),
  starts = "2 1",
  cov = c("independent", "zero"),
  k = c(1L, 3L),
  processors = 1L, run = TRUE,
  misstrick = TRUE, stringsAsFactors = FALSE)
tuneGrid$title <- paste0(
  c("linear", "quad", "spline")[sapply(tuneGrid$timevars, length)],
  "_",
  sapply(tuneGrid$cov, function(x) if(nchar(x)==4) substr(x, 1, 4) else substr(x, 1, 3)),
  "_",
  tuneGrid$k)
tuneGrid$base <- paste0(
  c("linear", "quad", "spline")[sapply(tuneGrid$timevars, length)],
  "_",
  sapply(tuneGrid$cov, function(x) if(nchar(x)==4) substr(x, 1, 4) else substr(x, 1, 3)))

## example using long2LGMM to fit one model at a time
mres <- long2LGMM(
  data = d,
  idvar = "ID",
  assessmentvar = "Assess",
  dv = tuneGrid$dv[1],
  timevars = tuneGrid$timevars[[1]],
  misstrick = tuneGrid$misstrick[1],
  k = tuneGrid$k[1],
  title = paste0(tuneGrid$title[1], tuneGrid$k[1]),
  base = tuneGrid$base[1],
  run = tuneGrid$run[1],
  processors = tuneGrid$processors[1],
  starts = tuneGrid$starts[1],
  newdata = nd,
  cov = tuneGrid$cov[1])

rm(mres)

## End(Not run)

```

lookupTech1Parameter *Lookup the matrix element for a give parameter number*

Description

The lookupTech1Parameter function identifies the position in the Mplus model matrices corresponding to a given parameter defined in the TECHNICAL 1 PARAMETER SPECIFICATION OUTPUT. The goal of this function is to aid in identifying problematic parameters often printed in the warnings and errors section of Mplus output.

Usage

```
lookupTech1Parameter(tech1Output, paramNumber)
```

Arguments

tech1Output	The object corresponding to the TECH1 parameter specification from readModels.
paramNumber	The parameter number to lookup

Value

A data.frame containing the row(s) and column(s) of TECH1 parameter specification matching the requested paramNumber.

Author(s)

Michael Hallquist

See Also

[readModels](#)

Examples

```
## Not run:  
models <- readModels("test1.out")  
param <- lookupTech1Parameter(models$tech1, 16)  
  
## End(Not run)
```

mplus.traceplot	<i>Plot the samples for each MCMC chain as a function of iterations</i>
-----------------	---

Description

Displays a traceplot of the MCMC draws from the poster distribution of each parameter estimate for a Bayesian Mplus model. This function requires that 1) PLOT: TYPE=PLOT2; be included in the Mplus input file, 2) a gh5 file be present corresponding to the Mplus output file (and containing a bayesian_data section), and 3) that the rhdf5 package be installed to allow the gh5 file to be imported.

Usage

```
mplus.traceplot(mplus.model, rows = 4, cols = 4, parameters_only = TRUE)
```

Arguments

mplus.model	An Mplus model extracted by the readModels function.
rows	Number of rows to display per plot.
cols	Optional. Number of columns to display per plot.
parameters_only	Optional. If TRUE, only the unstandardized parameter estimates from the MCMC draws will be displayed (as opposed to standardized estimates, r-square estimates, etc.). The unstandardized estimates all begin with "Parameter" in the Mplus gh5 output.

Details

A multi-panel plot is drawn to the screen and the user is prompted to display the next plot if more than rows x columns estimates are in the model.

Value

No value is returned by this function. Called for the side effect of displaying an MCMC chains traceplot.

Note

Trace and density plots can also be obtained using the coda package and the bparameters element of the mplus.model object. This requires that the posterior draws be saved using SAVEDATA: BPARAMETERS syntax. See example below.

Author(s)

Joseph Glass, Michael Hallquist

See Also[plot.mcmc](#)**Examples**

```
## Not run:
myModel <- readModels("BayesModel_WithGH5MCMC.out")
mplus.traceplot(myModel, rows=2, cols=3)

#alternative using the coda package
library(coda)
plot(myModel$bparameters$valid_draw)

## End(Not run)
```

MplusAutomation

Automating Mplus Model Estimation and Interpretation

Description

The MplusAutomation package leverages the flexibility of the R language to automate latent variable model estimation and interpretation using 'Mplus', a powerful latent variable modeling program developed by Muthen and Muthen (<http://www.statmodel.com>). Specifically, MplusAutomation provides routines for creating related groups of models, running batches of models, and extracting and tabulating model parameters and fit statistics.

Details

The MplusAutomation package has four primary purposes:

1. To automatically run groups/batches of models.
2. To provide routines to extract model fit statistics, parameter estimates, and raw data from 'Mplus' output files.
3. To facilitate comparisons among models
4. To provide a template language that allows for the creation of related input files.

The core routine for running batches of models is [runModels](#), with an easy-to-use GUI wrapper, [runModels_Interactive](#).

The core routine for extracting information from 'Mplus' outputs is [readModels](#), which returns a list containing all output sections that the package can extract.

To extract summaries, parameters, modification indices, SAVEDATA output, and all other sections that the package can understand, use the [readModels](#) function. This is the recommended way to extract 'Mplus' output with this package. If the `target` argument to [readModels](#) is a single .out file, an `mplus.model` (that is also a `list`) will be returned containing all output sections that the package can extract. If `target` is a directory, a list of `mplus.model` objects will be returned, named according to the output filenames.

Note: `extractModelSummaries` is deprecated and `readModels` should be preferred. To extract model summary statistics from one or more output files, see `extractModelSummaries`, which returns a `data.frame` of fit statistics for models located within a directory. Model fit results can be summarized in tabular form (for comparing among models) using `showSummaryTable` (displays table in separate window), `HTMLSummaryTable` (creates HTML file containing summary table), or `LatexSummaryTable` (returns a LaTeX-formatted table of summary statistics).

Deprecated: To extract raw data created by the `SAVEDATA` command (e.g., class membership probabilities or factor scores), see `getSavedata_Data`.

Deprecated: To extract unstandardized or standardized parameter estimates from a single output file, see `extractModelParameters`.

Detailed model fit and parameter comparisons between two models can be obtained using `compareModels`.

To create a group of related models from a single template, see `createModels`. Please read the `MplusAutomation` vignette provided along with the package (and on the CRAN website) in order to understand the template language: `vignette("Vignette", package="MplusAutomation")`.

In addition to the major functions above, a function for converting an R `data.frame` for use with 'Mplus' is provided: `prepareMplusData`. This converts the `data.frame` to a tab-delimited file and provides an 'Mplus' syntax stub for variable names.

```
Package:    MplusAutomation
Type:      Package
Version:   0.7-1
Date:     2018-01-31
License:   LGPL-3
LazyLoad: yes
```

Author(s)

Michael Hallquist <michael.hallquist@gmail.com>, Joshua F. Wiley <jwiley.psych@gmail.com>
 Maintainer: Michael Hallquist <michael.hallquist@gmail.com>

References

Mplus software. Muthen and Muthen. <http://www.statmodel.com>

See Also

See `runModels` for an example running a model.

mplusModeler

Create, run, and read Mplus models.

Description

This is a convenience wrapper to automate many of the usual steps required to run an Mplus model. It relies in part on functions from the `MplusAutomation` package.

Usage

```
mplusModeler(object, dataout, modelout, run = 0L, check = FALSE,
  varwarnings = TRUE, Mplus_command = "Mplus", writeData = c("ifmissing",
  "always", "never"), hashfilename = TRUE, ...)
```

Arguments

object	An object of class <code>mplusObject</code>
dataout	the name of the file to output the data to for Mplus. If missing, defaults to <code>modelout</code> changing <code>.inp</code> to <code>.dat</code> .
modelout	the name of the output file for the model. This is the file all the syntax is written to, which becomes the Mplus input file. It should end in <code>.inp</code> . If missing, defaults to <code>dataout</code> changing the extension to <code>.inp</code> .
run	an integer indicating how many models should be run. Defaults to zero. If zero, the data and model input files are all created, but the model is not run. This can be useful for seeing how the function works and what setup is done. If one, a basic model is run. If greater than one, the model is bootstrapped with <code>run</code> replications as well as the basic model.
check	logical whether the body of the Mplus syntax should be checked for missing semicolons using the <code>parseMplus</code> function. Defaults to <code>FALSE</code> .
varwarnings	A logical whether warnings about variable length should be left, the default, or removed from the output file.
Mplus_command	optional. N.B.: No need to pass this parameter for most users (has intelligent defaults). Allows the user to specify the name/path of the Mplus executable to be used for running models. This covers situations where Mplus is not in the system's path, or where one wants to test different versions of the Mplus program.
writeData	A character vector, one of 'ifmissing', 'always', 'never' indicating whether the data files (*.dat) should be written to disk. This is passed on to <code>prepareMplusData</code> . Note that previously, <code>mplusModeler</code> always (re)wrote the data to disk. However, now the default is to write the data to disk only if it is missing (i.e., 'ifmissing'). See details for further information.
hashfilename	A logical whether or not to add a hash of the raw data to the data file name. Defaults to <code>TRUE</code> in <code>mplusModeler</code> . Note that this behavior is a change from previous versions and differs from <code>prepareMplusData</code> which maintains the old behavior by default of <code>FALSE</code> .
...	additional arguments passed to the <code>prepareMplusData</code> function.

Details

Combined with functions from the `MplusAutomation` package, this function is designed to make it easy to fit Mplus models from R and to ease many of the usual frustrations with Mplus. For example, Mplus has very specific formats it accepts data in, but also very little data management facilities. Using R data management is easy. This function is designed to make using data from R in Mplus models easy. It is also common to want to fit many different models that are slight variants. This can be tedious in Mplus, but using R you can create one basic set of input, store it in a vector,

and then just modify that (e.g., using regular expressions) and pass it to Mplus. You can even use loops or the `*apply` constructs to fit the same sort of model with little variants.

The `writeData` argument is new and can be used to reduce overhead from repeatedly writing the same data from R to the disk. When using the ‘always’ option, `mplusModeler` behaves as before, always writing data from R to the disk. This remains the default for the `prepareMplusData` function to avoid confusion or breaking old code. However, for `mplusModeler`, the default has been set to ‘ifmissing’. In this case, R generates an md5 hash of the data prior to writing it out to the disk. The md5 hash is based on: (1) the dimensions of the dataset, (2) the variable names, (3) the class of every variable, and (4) the raw data from the first and last rows. This combination ensures that under most all circumstances, if the data changes, the hash will change. The hash is appended to the specified data file name (which is controlled by the logical `hashfilename` argument). Next R checks in the directory where the data would normally be written. If a data file exists in that directory that matches the hash generated from the data, R will use that existing data file instead of writing out the data again. A final option is ‘never’. If this option is used, R will not write the data out even if no file matching the hash is found.

Value

An Mplus model object, with results. If `run = 1`, returns an invisible list of results from the run of the Mplus model (see [readModels](#) from the `MplusAutomation` package). If `run = 0`, the function returns a list with two elements, ‘model’ and ‘boot’ that are both NULL. if `run >= 1`, returns a list with two elements, ‘model’ and ‘boot’ containing the regular Mplus model output and the boot object, respectively. In all cases, the Mplus data file and input files are created.

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

See Also

[runModels](#) and [readModels](#)

Examples

```
## Not run:
# minimal example of a model using builtin data, allowing R
# to automatically guess the correct variables to use
test <- mplusObject(MODEL = "mpg ON wt hp;
  wt WITH hp;", rdata = mtcars)

# estimate the model in Mplus and read results back into R
res <- mplusModeler(test, modelout = "model1.inp", run = 1L)

# when forcing writeData = "always" data gets overwritten (with a warning)
resb <- mplusModeler(test, modelout = "model1.inp", run = 1L,
  writeData = "always")

# using writeData = "ifmissing", the default, no data re-written
resc <- mplusModeler(test, modelout = "model1.inp", run = 1L)
```

```

# using writeData = "ifmissing", the default, data ARE written
# if data changes
test <- mplusObject(MODEL = "mpg ON wt hp;
  wt WITH hp;", rdata = mtcars[-10, ])
resd <- mplusModeler(test, modelout = "model1.inp", run = 1L)

# show summary
summary(resd)

# show coefficients
coef(resd)

# what if you wanted confidence intervals
# and standardized values?
# first update to tell Mplus you want them, re-run and print
test <- update(test, OUTPUT = ~ "CINTERVAL; STDYX;")
resd <- mplusModeler(test, modelout = "model1.inp", run = 1L)

coef(resd)
confint(resd)

# now standardized
coef(resd, type = "stdyx")
confint(resd, type = "stdyx")

# put together in one data frame if desired
merge(
  coef(resd, type = "stdyx"),
  confint(resd, type = "stdyx"),
  by = "Label")

# remove files
unlink(resc$results$input$data$file)
unlink(resd$results$input$data$file)
unlink("model1.inp")
unlink("model1.out")

# simple example of a model using builtin data
# demonstrates use with a few more sections
test2 <- mplusObject(
  TITLE = "test the MplusAutomation Package and mplusModeler wrapper;",
  MODEL = "
    mpg ON wt hp;
    wt WITH hp;",
  usevariables = c("mpg", "wt", "hp"),
  rdata = mtcars)

res2 <- mplusModeler(test2, modelout = "model2.inp", run = 1L)

# remove files
unlink(res2$results$input$data$file)
unlink("model2.inp")
unlink("model2.out")

```

```

# similar example using a robust estimator for standard errors
# and showing how an existing model can be easily updated and reused
test3 <- update(test2, ANALYSIS = ~ "ESTIMATOR = MLR;")

res3 <- mplusModeler(test3, modelout = "model3.inp", run = 1L)
unlink(res3$results$input$data$file)
unlink("model3.inp")
unlink("model3.out")

# now use the built in bootstrapping methods
# note that these work, even when Mplus will not bootstrap
# also note how categorical variables and weights are declared
# in particular, the usevariables for Mplus must be specified
# because more variables are included in the data than are in the
# model. Note the R usevariables includes all variables for both
# model and weights. The same is true for clustering.
test4 <- mplusObject(
  TITLE = "test bootstrapping;",
  VARIABLE = "
    CATEGORICAL = cyl;
    WEIGHT = wt;
    USEVARIABLES = cyl mpg;",
  ANALYSIS = "ESTIMATOR = MLR;",
  MODEL = "
    cyl ON mpg;",
  usevariables = c("mpg", "wt", "cyl"),
  rdata = mtcars)

res4 <- mplusModeler(test4, "mtcars.dat", modelout = "model4.inp", run = 10L,
  hashfilename = FALSE)
# see the results
res4$results$boot

# remove files
unlink("mtcars.dat")
unlink("model4.inp")
unlink("model4.out")

# Monte Carlo Simulation Example
montecarlo <- mplusObject(
  TITLE = "Monte Carlo Example;",
  MONTECARLO = "
    NAMES ARE i1-i5;
    NOBSERVATIONS = 100;
    NREPS = 100;
    SEED = 1234;",
  MODELPOPULATION = "
    f BY i1-i5*1;
    f@1;
    i1-i5*1;",
  ANALYSIS = "
    ESTIMATOR = BAYES;

```

```

PROC = 2;
fbiter = 100;";
MODEL = "
f BY i1-i5*.8 (11-15);
f@1;
i1-i5*1;";
MODELPRIORS = "
l1-l5 ~ N(.5 .1);";
OUTPUT = "TECH9;";

fitMonteCarlo <- mplusModeler(montecarlo,
  modelout = "montecarlo.inp",
  run = 1L,
  writeData = "always",
  hashfilename = FALSE)

unlink("montecarlo.inp")
unlink("montecarlo.out")

# Example including ID variable and extracting factor scores
dat <- mtcars
dat$UID <- 1:nrow(mtcars)

testIDs <- mplusObject(
  TITLE = "test the mplusModeler wrapper with IDs;",
  VARIABLE = "IDVARIABLE = UID;",
  MODEL = "
  F BY mpg wt hp;";
  SAVEDATA = "
  FILE IS testid_fscores.dat;
  SAVE IS fscores;
  FORMAT IS free;";
  usevariables = c("UID", "mpg", "wt", "hp"),
  rdata = dat)

resIDs <- mplusModeler(testIDs, modelout = "testid.inp", run = 1L)

# view the saved data from Mplus, including factor scores
# the indicator variables, and the ID variable we specified
head(resIDs$results$savedata)

# merge the factor scores with the rest of the original data
# merge together by the ID column
dat <- merge(dat, resIDs$results$savedata[, c("F", "UID")],
  by = "UID")

# correlate merged factor scores against some other new variable
with(dat, cor(F, qsec))

# remove files
unlink(resIDs$results$input$data$file)
unlink("testid.inp")

```

```

unlink("testid.out")
unlink("testid_fscores.dat")
unlink("Mplus Run Models.log")

## End(Not run)

```

mplusObject

Create an Mplus model object

Description

This is a function to create an Mplus model object in R. The object holds all the sections of an Mplus input file, plus some extra R ones. Once created, the model can be run using other functions such as `mplusModeler` or updated using methods defined for the update function.

Usage

```

mplusObject(TITLE = NULL, DATA = NULL, VARIABLE = NULL, DEFINE = NULL,
  MONTECARLO = NULL, MODELPOPULATION = NULL, MODELMISSING = NULL,
  ANALYSIS = NULL, MODEL = NULL, MODELINDIRECT = NULL,
  MODELCONSTRAINT = NULL, MODELTEST = NULL, MODELPRIORS = NULL,
  OUTPUT = NULL, SAVEDATA = NULL, PLOT = NULL, usevariables = NULL,
  rdata = NULL, autov = TRUE, imputed = FALSE)

```

Arguments

TITLE	A character string of the title for Mplus.
DATA	A character string of the data section for Mplus (note, do not define the filename as this is generated automatically)
VARIABLE	A character string of the variable section for Mplus (note, do not define the variable names from the dataset as this is generated automatically)
DEFINE	A character string of the define section for Mplus (optional)
MONTECARLO	A character string of the montecarlo section for Mplus (optional). If used, <code>autov</code> defaults to FALSE instead of the usual default, TRUE, but may still be overwritten, if desired.
MODELPOPULATION	A character string of the MODEL POPULATION section for Mplus (optional).
MODELMISSING	A character string of the MODEL MISSING section for Mplus (optional).
ANALYSIS	A character string of the analysis section for Mplus (optional)
MODEL	A character string of the model section for Mplus (optional, although typically you want to define a model)
MODELINDIRECT	A character string of the MODEL INDIRECT section for Mplus (optional).
MODELCONSTRAINT	A character string of the MODEL CONSTRAINT section for Mplus (optional).

MODELTEST	A character string of the MODEL TEST section for Mplus (optional).
MODELPRIORS	A character string of the MODEL PRIORS section for Mplus (optional).
OUTPUT	A character string of the output section for Mplus (optional)
SAVEDATA	A character string of the savedata section for Mplus (optional)
PLOT	A character string of the plot section for Mplus (optional)
usevariables	A character vector of the variables from the R dataset to use in the model.
rdata	An R dataset to be used for the model.
autov	A logical (defaults to TRUE) argument indicating whether R should attempt to guess the correct variables to use from the R dataset, if usevariables is left NULL.
imputed	A logical whether the data are multiply imputed (a list). Defaults to FALSE.

Details

Mplus model objects allow a base model to be defined, and then flexibly update the data, change the precise model, etc. If a section does not vary between models, you can leave it the same. For example, suppose you are fitting a number of models, but in all cases, wish to use maximum likelihood estimator, “ANALYSIS: ESTIMATOR = ML;” and would like standardized output, “OUTPUT: STDYX;”. Rather than retype those in every model, they can be defined in one Mplus model object, and then that can simply be updated with different models, leaving the analysis and output sections untouched. This also means that if a reviewer comes back and asks for all analyses to be re-run say using the robust maximum likelihood estimator, all you have to do is change it in the model object once, and re run all your code.

Value

A list of class `mplusObject` with elements

TITLE	The title in Mplus (if defined)
DATA	The data section in Mplus (if defined)
VARIABLE	The variable section in Mplus (if defined)
DEFINE	The define section in Mplus (if defined)
MONTECARLO	The montecarlo section in Mplus (if defined)
MODELPOPULATION	The modelpopulation section in Mplus (if defined)
MODELMISSING	The modelmissing section in Mplus (if defined)
ANALYSIS	The analysis section in Mplus (if defined)
MODEL	The model section in Mplus (if defined)
MODELINDIRECT	The modelindirect section in Mplus (if defined)
MODELCONSTRAINT	The modelconstraint section in Mplus (if defined)
MODELTEST	The modeltest section in Mplus (if defined)
MODELPRIORS	The modelpriors section in Mplus (if defined)

OUTPUT	The output section in Mplus (if defined)
SAVEDATA	The savedata section in Mplus (if defined)
PLOT	The plot section in Mplus (if defined)
results	NULL by default, but can be later updated to include the results from the model run.
usevariables	A character vector of the variables from the R data set to be used.
rdata	The R data set to use for the model.
imputed	A logical whether the data are multiply imputed.
autov	A logical whether the data should have the usevariables detected automatically or not

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

See Also

[mplusModeler](#)

Examples

```
example1 <- mplusObject(MODEL = "mpg ON wt;",
  usevariables = c("mpg", "hp"), rdata = mtcars)
str(example1)
rm(example1)

# R figures out the variables automagically, with a message
example2 <- mplusObject(MODEL = "mpg ON wt;",
  rdata = mtcars, autov = TRUE)
str(example2)
rm(example2)

# R can also try to figure out a list of variables when
# variable names are hyphenated first-last variable, all variables
# between the first and last one will be included
example3 <- mplusObject(MODEL = "mpg ON wt-vs;",
  rdata = mtcars, autov = TRUE)
str(example3)
rm(example3)

# R warns if the first 8 characters of a (used) variable name are not unique
# as they will be indistinguishable in the Mplus output
example4 <- mplusObject(MODEL = "baseline_01 ON baseline_02;",
  rdata = data.frame(baseline_01 = 1:5, baseline_02 = 5:1),
  autov = TRUE)
rm(example4)
```

mplusRcov

Create Mplus code for various residual covariance structures.

Description

This function makes it easy to write the Mplus syntax for various residual covariance structure.

Usage

```
mplusRcov(x, type = c("homogenous", "heterogenous", "cs", "toeplitz", "ar",
  "un"), r = "rho", e = "e", collapse = FALSE)
```

Arguments

x	input character vector of variable names, ordered by time
type	A character string indicating the type of residual covariance structure to be used. Defaults to 'homogenous'. Current options include 'homogenous', 'heterogenous', 'cs' for compound symmetric, 'toeplitz' for banded toeplitz, 'ar' for autoregressive, and 'un' for unstructured.
r	a character vector of the base label to name covariance parameters. Defaults to 'rho'.
e	a character vector of the error variance of the variable. Used to create constraints on the covariance parameters. Defaults to 'e'.
collapse	whether to collapse the covariance code using 'PWITH'. Note that at the time of writing, Mplus does not allow more than 80 characters per row. Defaults to FALSE.

Details

The **homogenous** residual covariance structure estimates one parameter: the residual variance, σ_e^2 . The residual variance is assumed to be identical for all variables and all covariances are assumed to be zero. The structure is represented in this table.

	t1	t2	t3	...	tn
t1	σ_e^2			...	
t2	0	σ_e^2		...	
t3	0	0	σ_e^2	...	
...
tn	0	0	0	...	σ_e^2

The **heterogenous** residual covariance structure estimates **n** parameters, where **n** is the number of variables. A unique residual variance is estimated for every variable. All covariances are assumed to be zero. The structure is represented in this table.

t1	t2	t3	...	tn
----	----	----	-----	----

t1	σ_{e1}^2			...
t2	0	σ_{e2}^2		...
t3	0	0	σ_{e3}^2	...
...
tn	0	0	0	... σ_{en}^2

The **compound symmetric** residual covariance structure estimates two parameters: one for the residual variance, σ_e^2 , and one for the covariance. The residual variance is assumed to be identical for all variables and all covariances are assumed to be identical. The structure is represented in this table.

	t1	t2	t3	...	tn
t1	σ_e^2			...	
t2	ρ	σ_e^2		...	
t3	ρ	ρ	σ_e^2	...	
...
tn	ρ	ρ	ρ	...	σ_e^2

The **toeplitz** residual covariance structure estimates **n** parameters, one for every band of the matrix. The residual variance, σ_e^2 , is assumed to be identical for all variables. The covariances one step removed are all assumed identical. Likewise for all further bands. The structure is represented in this table.

	t1	t2	t3	...	tn
t1	σ_e^2			...	
t2	ρ	σ_e^2		...	
t3	ρ_2	ρ	σ_e^2	...	
...
tn	ρ_n	ρ_{n-1}	ρ_{n-2}	...	σ_e^2

The **autoregressive** residual covariance structure has two parameters: the residual variance, σ_e^2 and the correlation between adjacent time points, ρ . The variances are constrained to be equal for all time points. A single correlation parameter is estimated. The ρ is the correlation between adjacent time points such as 1 and 2 or 2 and 3. More distant relationships are assumed to have smaller correlations, decreasing exponentially. Thus between 1 and 3, the estimate is ρ^2 . The structure is represented in this table.

	t1	t2	t3	...	tn
t1	σ_e^2			...	
t2	ρ	σ_e^2		...	
t3	ρ^2	ρ	σ_e^2	...	
...
tn	ρ^{n-1}	ρ^{n-2}	ρ^{n-3}	...	σ_e^2

Because structural equation models generally model covariance structures, the autoregressive residual structure must be parameterized in terms of covariances. This is done in two parts. First, the function returns syntax to estimate all the pairwise covariances, labelling the parameters ρ , ρ^2 , etc. so that they are constrained to be equal. Next, it returns the syntax for the necessary model constraints to constrain the different covariances, to decrease exponentially in their correlations. This is done via:

$$\rho^2 = \left(\frac{\rho}{\sigma_e^2}\right)^2 \sigma_e^2$$

and likewise for all later time points.

The **unstructured** residual covariance structure estimates $\frac{n(n+1)}{2}$ parameters. It is unstructured in that every variance and covariance is freely estimated with no constraints. However, in most cases, this results in an overparameterized model and is unestimable. The structure is represented in this table.

	t1	t2	t3	...	tn
t1	σ_{e1}^2			...	
t2	ρ_1	σ_{e2}^2		...	
t3	ρ_2	ρ_3	σ_{e3}^2	...	
...
tn	ρ_5	ρ_6	ρ_7	...	σ_{en}^2

Value

A named character vector of class 'MplusRstructure' with four elements:

all	A character string collapsing all other sections.
Variances	A character string containing all of the variances.
Covariances	A character string containing all of the covariances, properly labelled to allow constraints and the autoregressive residual covariance structure.
Constraints	A character string containing the 'MODEL CONSTRAINT' section and code needed to parameterize the residual covariance structure as autoregressive.

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

Examples

```
# all five structures collapsing
mplusRcov(letters[1:4], "homogenous", "rho", "e", TRUE)
mplusRcov(letters[1:4], "heterogenous", "rho", "e", TRUE)
mplusRcov(letters[1:4], "cs", "rho", "e", TRUE)
mplusRcov(letters[1:4], "toeplitz", "rho", "e", TRUE)
mplusRcov(letters[1:4], "ar", "rho", "e", TRUE)
mplusRcov(letters[1:4], "un", "rho", "e", TRUE)

# all five structures without collapsing
# useful for long names or many variables
```

```
# where a line may cross 80 characters
mplusRcov(letters[1:4], "homogenous", "rho", "e", FALSE)
mplusRcov(letters[1:4], "heterogenous", "rho", "e", FALSE)
mplusRcov(letters[1:4], "cs", "rho", "e", FALSE)
mplusRcov(letters[1:4], "toeplitz", "rho", "e", FALSE)
mplusRcov(letters[1:4], "ar", "rho", "e", FALSE)
mplusRcov(letters[1:4], "un", "rho", "e", FALSE)
```

paramExtract

Extract parameters from a data frame of Mplus estimates

Description

This is a simple convenience function designed to facilitate looking at specific parameter types by easily return a subset of a data frame with those types only. It is designed to follow up the results returned from the [readModels](#) function.

Usage

```
paramExtract(x, params = c("regression", "loading", "undirected",
  "expectation", "variability", "new"))
```

Arguments

x	A data frame (specifically the type returned by <code>readModels</code>) containing parameters. Should be specific such as unstandardized and the data frame must have a column called 'paramHeader'.
params	A character string indicating the types of parameters to be returned. Options currently include 'regression', 'loading', 'undirected', 'expectation', 'variability', and 'new' for new/additional parameters. Regressions include regression of one variable ON another. 'loading' include indicator variables (which are assumed caused by the underlying latent variable) and variables in latent growth models (BY or I). Undirected paths currently only include covariances, indicated by the WITH syntax in Mplus. Expectation paths are the unconditional or conditional expectations of variables. In other words those parameters related to the first moments. For independent variables, these are the means, $E(X)$ and the conditional means or intercepts, $E(X f(\theta))$ where $f(\theta)$ is the model, some function of the parameters, θ . Finally 'variability' refers to both variances and residual variances, corresponding to the second moments. As with the expectations, variances are unconditional for variables that are not predicted or conditioned on any other variable in the model whereas residual variances are conditional on the model. Note that R uses fuzzy matching so that each of these can be called via shorthand, 'r', 'l', 'u', 'e', and 'v'.

Value

A subset data frame with the parameters of interest.

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

See Also

[readModels](#)

Examples

```
## Not run:
test <- mplusObject(
  TITLE = "test the MplusAutomation Package and my Wrapper;",
  MODEL = "
    mpg ON wt hp;
    wt WITH hp;",
  usevariables = c("mpg", "wt", "hp"),
  rdata = mtcars)

res <- mplusModeler(test, "mtcars.dat", modelout = "model1.inp", run = 1L)

# store just the unstandardized parameters in 'd'
d <- res$results$parameters$unstandardized
# extract just regression parameters
paramExtract(d, "regression")
# extract other types of parameters using shorthand
paramExtract(d, "u")
paramExtract(d, "e")
paramExtract(d, "v")

## End(Not run)
```

parseCatOutput

Parse Categorical Output

Description

Helper function for parsing output with variables and categories.

Usage

```
parseCatOutput(text)
```

Arguments

text The output to parse.

Value

The parsed output

Author(s)

Michael Hallquist

Examples

```

"
Example:
UNIVARIATE PROPORTIONS AND COUNTS FOR CATEGORICAL VARIABLES

SOP2A
  Category 1   0.254   631.000
  Category 2   0.425  1056.000
  Category 3   0.174   432.000
  Category 4   0.147   365.000

Or Item Categories in IRT Parameterization

Item Categories
U1
  Category 1   0.000   0.000   0.000   1.000
  Category 2  -0.247   0.045  -5.534   0.000
  Category 3   0.699   0.052  13.325   0.000
  Category 4  -0.743   0.057 -12.938   0.000
  Category 5   0.291   0.052   5.551   0.000
"

```

 parseMplus

Check Mplus code for missing semicolons or too long lines.

Description

The function parses a character string containing Mplus code and checks that every non blank line ends in either a colon or a semicolon. In addition, it checks that every line is less than 90 characters, because Mplus ignores everything after 90 characters on a line which can be a source of enigmatic errors.

Usage

```
parseMplus(x, add = FALSE)
```

Arguments

x a character string containing Mplus code.

add logical indicating whether or not to add semicolons to lines that do not have them. Defaults to FALSE.

Details

The function is fairly basic at the moment. It works by simply removing blank space (spaces, tabs, etc.) and then if a line does not terminate in a colon or semicolon, it returns a note and the line number. Optionally, it can add semicolons to any lines missing them and return the input with added semicolons. To check for lines that are too long, all trailing (but not before) white space is removed, and then the number of characters is checked.

Value

a character vector containing the input text and optionally added semicolons.

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

See Also

[mplusModeler](#)

Examples

```
# sample input
test <- "
MODEL:
  mpg ON wt hp;
  wt WITH hp
"
# check and return
cat(parseMplus(test), fill=TRUE)
# add missing semicolons and return
cat(parseMplus(test, TRUE), fill=TRUE)
test <- "
MODEL:
  mpg cyl disp hp drat wt qsec vs am gear PWITH cyl disp hp drat wt qsec vs am gear carb;
"
cat(parseMplus(test))
```

plot.mplusObject

Plot coefficients for an mplusObject

Description

This is a method for plotting the coefficients of an mplusObject.

Usage

```
## S3 method for class 'mplusObject'
plot(x, y, type = c("stdyx", "un", "std", "stdy"), ...)
```


Arguments

x	An object of class mplusObject
y	Not currently used
type	A character vector indicating the type of coefficients to return. One of “un”, “std”, “stdy”, or “stdyx”. Defaults to “stdyx”.
...	Additional arguments to pass on (not currently used)

Value

Nothing. Called for its side effect of plotting the coefficients.

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

Examples

```
## Not run:
# simple example of a model using builtin data
# demonstrates use
test <- mplusObject(
  TITLE = "test the MplusAutomation Package;",
  MODEL = "
    mpg ON wt hp;
    wt WITH hp;",
  OUTPUT = "STANDARDIZED;",
  usevariables = c("mpg", "wt", "hp"),
  rdata = mtcars)

res <- mplusModeler(test, "mtcars.dat", modelout = "model1.inp", run = 1L)

# example of the coef method
plot(res)

# remove files
unlink("mtcars.dat")
unlink("model1.inp")
unlink("model1.out")
unlink("Mplus Run Models.log")

## End(Not run)
```

Description

The prepareMplusData function converts an R data.frame object into a tab-delimited file (without header) to be used in an Mplus input file. The corresponding Mplus syntax, including the data file definition and variable names, is printed to the console or optionally to an input file.

Usage

```
prepareMplusData(df, filename, keepCols, dropCols, infile = FALSE,
  interactive = TRUE, overwrite = TRUE, imputed = FALSE,
  writeData = c("always", "ifmissing", "never"), hashfilename = FALSE)
```

Arguments

df	The R data.frame to be prepared for Mplus
filename	The path and filename for the tab-delimited data file for use with Mplus. Example: "C:/Mplusdata/data1.dat"
keepCols	A character vector specifying the variable names within df to be output to filename or a numeric vector of the column indices to be output or a logical vector corresponding to the same.
dropCols	A character vector specifying the variable names within df to be omitted from the data output to filename or a numeric vector of the column indices not to be output or a logical vector corresponding to the same.
infile	Logical value whether the Mplus syntax should be written to the console or to an input file. Defaults to FALSE. If TRUE, the file name will be the same as filename with the extension changed to .inp. Alternately, this can be a character string giving the file name to write the Mplus syntax to.
interactive	Logical value indicating whether file names should be selected interactively. If filename is missing and interactive=TRUE, then a dialogue box will pop up to select a file or a console prompt if in a non interactive context. Defaults to TRUE.
overwrite	Logical value indicating whether data and input (if present) files should be overwritten. Defaults to TRUE to be consistent with prior behavior. If FALSE and the file to write the data to already exists, it will throw an error.
imputed	A logical whether data are multiply imputed. Defaults to FALSE. If TRUE, the data should be a list, where each element of the list is a multiply imputed dataset.
writeData	A character vector, one of 'always', 'ifmissing', 'never' indicating whether the data files (*.dat) should be written to disk. Defaults to 'always' for consistency with previous behavior. See details for further information.
hashfilename	A logical whether or not to add a hash of the raw data to the data file name. Defaults to FALSE for consistency with previous behavior where this feature was not available..

Details

The writeData argument is new and can be used to reduce overhead from repeatedly writing the same data from R to the disk. When using the 'always' option, prepareMplusData behaves as before, always writing data from R to the disk. When 'ifmissing', R generates an md5 hash of

the data prior to writing it out to the disk. The md5 hash is based on: (1) the dimensions of the dataset, (2) the variable names, (3) the class of every variable, and (4) the raw data from the first and last rows. This combination ensures that under most all circumstances, if the data changes, the hash will change. The hash is appended to the specified data file name (which is controlled by the logical `hashfilename` argument). Next R checks in the directory where the data would normally be written. If a data file exists in that directory that matches the hash generated from the data, R will use that existing data file instead of writing out the data again. A final option is 'never'. If this option is used, R will not write the data out even if no file matching the hash is found.

Value

Invisibly returns a character vector of the Mplus input syntax. Primarily called for its side effect of creating Mplus data files and optionally input files.

Author(s)

Michael Hallquist

Examples

```
## Not run:
library(foreign)

study5 <- read.spss("reanalysis-study-5-mt-fall-08.sav", to.data.frame=TRUE)
ASData5 <- subset(study5, select=c("ppnum", paste("as", 1:33, sep="")))

prepareMplusData(ASData5, "study5.dat")

# basic example
test01 <- prepareMplusData(mtcars, "test01.dat")

# see that syntax was stored
test01

# by default, if re-run, data is re-written, with a note
test01b <- prepareMplusData(mtcars, "test01.dat")

# if we turn on hashing in the filename the first time,
# we can avoid overwriting notes the second time
test01c <- prepareMplusData(mtcars, "test01c.dat", hashfilename=TRUE)

# now that the filename was hashed in test01c, future calls do not re-write data
# as long as the hash matches
test01d <- prepareMplusData(mtcars, "test01c.dat", writeData = "ifmissing", hashfilename=TRUE)

# however, if the data change, then the file is re-written
test01e <- prepareMplusData(iris, "test01c.dat", writeData = "ifmissing", hashfilename=TRUE)

# tests for keeping and dropping variables
prepareMplusData(mtcars, "test02.dat", keepCols = c("mpg", "hp"))
prepareMplusData(mtcars, "test03.dat", keepCols = c(1, 2))
```

```

prepareMplusData(mtcars, "test04.dat",
  keepCols = c(TRUE, FALSE, FALSE, TRUE, FALSE,
    FALSE, FALSE, FALSE, FALSE, FALSE, FALSE))

prepareMplusData(mtcars, "test05.dat", dropCols = c("mpg", "hp"))
prepareMplusData(mtcars, "test06.dat", dropCols = c(1, 2))
prepareMplusData(mtcars, "test07.dat",
  dropCols = c(TRUE, FALSE, FALSE, TRUE, FALSE,
    FALSE, FALSE, FALSE, FALSE, FALSE, FALSE))

# interactive (test08.dat)
prepareMplusData(mtcars, interactive=TRUE)

# write syntax to input file, not stdout
prepareMplusData(mtcars, "test09.dat", infile=TRUE)

# write syntax to alternate input file, not stdout
prepareMplusData(mtcars, "test10.dat", infile="test10alt.inp")

# should be error, no file
prepareMplusData(mtcars, interactive=FALSE)

# new warnings if it is going to overwrite files
# (the default to be consistent with prior behavior)
prepareMplusData(mtcars, "test10.dat")

# new warnings if it is going to overwrite files
# (the default to be consistent with prior behavior)
prepareMplusData(mtcars, "test11.dat", infile="test10alt.inp")

# new errors if files exist and overwrite=FALSE
prepareMplusData(mtcars, "test10.dat",
  infile="test10alt.inp", overwrite=FALSE)

## End(Not run)

```

`print.MplusRstructure` *Print an Mplus Residual Structure object*

Description

This is a method for printing an Mplus Residual Structure object.

Usage

```

## S3 method for class 'MplusRstructure'
print(x, ...)

```

Arguments

x An object of class MplusRstructure
... Additional arguments to pass on (not currently used)

Value

NULL Called for its side effect of printing the object to the console

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

See Also

Other Mplus-Formatting: [coef.mplus.model](#), [confint.mplus.model](#), [extract](#), [summary.mplusObject](#)

Examples

```
# default 'show' uses printing
mplusRcov(c("a", "b", "c"), type = "ar")

# also if calling print explicitly
print(mplusRcov(c("a", "b", "c"), type = "ar"))

# to see all aspects of the raw/original object
str(mplusRcov(c("a", "b", "c"), type = "ar"))
```

readModels	<i>Read Parameters, Summary Statistics, and Savedata from Mplus Output</i>
------------	--

Description

Extracts information from one or more Mplus output files, including fit statistics and parameters. Its is to parse all (supported) aspects of Mplus output and to combine these into a list object, with one element per output file identified.

Usage

```
readModels(target = getwd(), recursive = FALSE, filefilter, what = "all",
  quiet = FALSE)
```

Arguments

target	the directory containing Mplus output files (.out) to parse OR the single output file to be parsed. May be a full path, relative path, or a filename within the working directory. Defaults to the current working directory. Example: "C:/Users/Michael/Mplus Runs"
recursive	optional. If TRUE, parse all models nested in subdirectories within target. Defaults to FALSE.
filefilter	a Perl regular expression (PCRE-compatible) specifying particular output files to be parsed within directory. See regex or http://www.pcre.org/pcre.txt for details about regular expression syntax.
what	a character vector denoting what aspects of Mplus output to extract. Defaults to "all", which will extract all supported output sections. See details for additional information.
quiet	whether to suppress printing to the screen the file currently being processed. Defaults to FALSE.

Details

The what parameter defaults to "all", which extracts all supported output. If you would like to extract a reduced set of output sections (especially to speed up the function when reading many files), specify the sections as a character vector from the following options:

```
c("input", "warn_err", "sampstat", "covariance_coverage", "summaries", "parameters", "class_counts",
  "indirect", "mod_indices", "residuals", "savedata", "bparameters", "tech1", "tech3", "tech4", "tech7",
  "tech8", "tech9", "tech12", "fac_score_stats", "lcCondMeans", "gh5")
```

Value

A list with one mplus.model per file. Each mplus.model object is composed of elements containing major output sections, as detailed below. If target is a single file, then the top-level elements will be a single mplus.model object, not a list of files. Specific elements are:

input	Mplus input syntax parsed into a list by major section
warnings	Syntax and estimation warnings as a list
errors	Syntax and estimation errors as a list
sampstat	Sample statistics provided by OUTPUT: SAMPSTAT, if specified
covariance_coverage	Covariance coverage matrix for checking missingness patterns
summaries	Summary statistics from extractModelSummaries, having structure as specified by that function
parameters	Model parameters from extractModelParameters, having structure as specified by that function
class_counts	Latent class counts and proportions for models that include a categorical latent variable
indirect	Output of MODEL INDIRECT if available in output. Contains \$overall and \$specific data.frames for indirect effects

mod_indices	Model modification indices from extractModIndices, having structure as specified by that function
residuals	a list containing relevant information from OUTPUT: RESIDUALS
savedata_info	File information about SAVEDATA files related to this output
savedata	SAVEDATA file as an R data.frame, as described in getSavedata_Data
bparameters	an mcmc.list object containing the draws from the MCMC chains for a Bayesian model that uses the SAVEDATA: BPARAMETERS command
tech1	a list containing parameter specification and starting values from OUTPUT: TECH1
tech3	a list containing parameter covariance and correlation matrices from OUTPUT: TECH3
tech4	a list containing means, covariances, and correlations for latent variables from OUTPUT: TECH4
tech7	a list containing sample statistics for each latent class from OUTPUT: TECH7
tech8	a list containing optimization history of the model. Currently only supports potential scale reduction in BAYES. OUTPUT: TECH8
tech9	a list containing warnings/errors from replication runs for MONTECARLO analyses from OUTPUT: TECH9
tech12	a list containing observed versus estimated sample statistics for TYPE=MIXTURE analyses from OUTPUT: TECH12
fac_score_stats	factor score mean, correlation, and covariance structure from SAMPLE STATISTICS FOR ESTIMATED FACTOR SCORES section
lcCondMeans	conditional latent class means and pairwise comparisons, obtained using auxiliary(e) syntax in latent class models
gh5	a list containing data from the gh5 (graphics) file corresponding to this output. (Requires rhdf5 package)

Author(s)

Michael Hallquist

Examples

```
## Not run:
  allOutput <- readModels(
    "C:/Program Files/Mplus/Mplus Examples/User's Guide Examples", recursive=TRUE)

## End(Not run)
```

runModels

*Run Mplus Models***Description**

This function runs a group of Mplus models (.inp files) located within a single directory or nested within subdirectories.

Usage

```
runModels(target = getwd(), recursive = FALSE, filefilter = NULL,
  showOutput = FALSE, replaceOutfile = "always",
  logFile = "Mplus Run Models.log", Mplus_command = "Mplus",
  killOnFail = TRUE, local_tmpdir = FALSE)
```

Arguments

target	the directory containing Mplus input files (.inp) to run OR the single .inp file to be run. May be a full path, relative path, or a filename within the working directory. Defaults to the current working directory. Example: "C:/Users/Michael/Mplus Runs"
recursive	optional. If TRUE, run all models nested in subdirectories within directory. Defaults to FALSE. Not relevant if target is a single file.
filefilter	a Perl regular expression (PCRE-compatible) specifying particular input files to be run within directory. See regex or http://www.pcre.org/pcre.txt for details about regular expression syntax. Not relevant if target is a single file.
showOutput	optional. If TRUE, show estimation output (TECH8) in the R console. Note that if run within Rgui, output will display within R, but if run via Rterm, a separate window will appear during estimation.
replaceOutfile	optional. Currently supports three settings: "always", which runs all models, regardless of whether an output file for the model exists; "never", which does not run any model that has an existing output file; and "modifiedDate", which only runs a model if the modified date for the input file is more recent than the output file modified date (implying there have been updates to the model).
logFile	optional. If non-null, specifies a file (and optionally, directory) that records the settings passed into the function and the models run (or skipped) during the run.
Mplus_command	optional. N.B.: No need to pass this parameter for most users (has intelligent defaults). Allows the user to specify the name/path of the Mplus executable to be used for running models. This covers situations where Mplus is not in the system's path, or where one wants to test different versions of the Mplus program.
killOnFail	optional. Windows only for now. If TRUE, kill all processes named mplus.exe when runModels does not terminate normally. Defaults to TRUE.

local_tmpdir optional. Linux/Mac for now. If TRUE, set the TMPDIR environment variable to the location of the .inp file prior to execution. This is useful in Monte Carlo studies where many instances of Mplus may run in parallel and we wish to avoid collisions in temporary files among processes.

Value

None. Function is used for its side effects (running models).

Author(s)

Michael Hallquist

See Also

[runModels_Interactive](#)

Examples

```
## Not run:
runModels("C:/Users/Michael/Mplus Runs", recursive=TRUE, showOutput=TRUE,
  replaceOutfile="modifiedDate", logFile="MH_RunLog.txt",
  Mplus_command="C:\\Users\\Michael\\Mplus Install\\Mplus51.exe")

## End(Not run)
```

runModels_Interactive *Run Mplus Models Using Graphical Interface*

Description

This function provides a graphical user interface to the runModels function. It uses Tcl/Tk to display a window in which the user can specify parameters for runModels, including the directory for runs, recursing through subdirectories, displaying output on the console, and replacing existing outfiles.

Usage

```
runModels_Interactive(directory = getwd(), recursive = "0",
  showOutput = "1", replaceOutfile = "1", checkDate = "0",
  logFile = "1")
```

Arguments

directory optional. The starting directory that will display in the dialog window. Defaults to the current working directory.

recursive optional. Whether the recursive checkbox should be checked when the window opens. "0" for FALSE, "1" for TRUE.

showOutput	optional. Whether the show output checkbox should be checked when the window opens. “0” for FALSE, “1” for TRUE.
replaceOutfile	optional. Whether the replace outfile checkbox should be checked when the window opens. “0” for FALSE, “1” for TRUE.
checkDate	optional. Whether the check modified date checkbox should be checked when the window opens. “0” for FALSE, “1” for TRUE.
logFile	optional. Whether the log file checkbox should be checked when the window opens. “0” for FALSE, “1” for TRUE.

Details

This function exists as a GUI wrapper for `runModels` and does not provide any distinct functionality.

Value

None. Function is used to display user interface for running models.

Author(s)

Michael Hallquist

See Also

[runModels](#)

Examples

```
# interactive, none
```

separateHyphens	<i>Separate Hyphenated Variable Strings</i>
-----------------	---

Description

This code is a simplified form of `expandCmd` from the **lavaan** package. It separates hyphenated variable strings into a list of vectors, while ignoring hyphens that may be used in numbers.

Usage

```
separateHyphens(cmd)
```

Arguments

cmd	A character string
-----	--------------------

Details

Note that this is an internal function only.

Value

The character string if no hyphens, or a list of vectors if there are hyphens.

Author(s)

Michael Hallquist revised by Joshua Wiley

Examples

```
MplusAutomation:::separateHyphens("x1x4")
MplusAutomation:::separateHyphens("x1-x4")
MplusAutomation:::separateHyphens("x1-x4; x1*-1; v1-v3;")
```

showSummaryTable	<i>Display summary table of Mplus model statistics in separate window</i>
------------------	---

Description

Displays a summary table of model fit statistics extracted using the `extractModelSummaries` function. This function relies on the `showData` function from the `relimp` package, which displays data in a Tk-based window. By default, the following summary statistics are included: Title, LL, Parameters, AIC, AICC, BIC, RMSEA_Estimate, but these are customizable using the `keepCols` and `dropCols` parameters.

Usage

```
showSummaryTable(modelList, keepCols, dropCols, sortBy, font = "Courier 9")
```

Arguments

<code>modelList</code>	A list of models (as a <code>data.frame</code>) returned from the <code>extractModelSummaries</code> function.
<code>keepCols</code>	A vector of character strings indicating which columns/variables to display in the summary. Only columns included in this list will be displayed (all others excluded). By default, <code>keepCols</code> is: <code>c("Title", "LL", "Parameters", "AIC", "AICC", "BIC", "RMSEA_Estimate")</code> . Example: <code>c("Title", "LL", "AIC", "CFI")</code>
<code>dropCols</code>	A vector of character strings indicating which columns/variables to omit from the summary. Any column not included in this list will be displayed. By default, <code>dropCols</code> is <code>NULL</code> . Example: <code>c("InputInstructions", "TLI")</code>
<code>sortBy</code>	Optional. Field name (as character string) by which to sort the table. Typically an information criterion (e.g., "AIC" or "BIC") is used to sort the table. Defaults to "AICC".
<code>font</code>	Optional. The font to be used to display the summary table. Defaults to Courier 9.

Value

No value is returned by this function. It is solely used to display the summary table in a separate window.

Note

You must choose between keepCols and dropCols because it is not sensible to use these together to include and exclude columns. The function will error if you include both parameters.

Author(s)

Michael Hallquist

See Also

[extractModelSummaries](#) [HTMLSummaryTable](#) [LatexSummaryTable](#)

Examples

```
# make me!!!
```

summary.mplusObject *Summarize an mplusObject*

Description

This is a method for summarizing an mplusObject.

Usage

```
## S3 method for class 'mplusObject'  
summary(object, verbose = FALSE, ...)
```

Arguments

object	An object of class mplusObject
verbose	Logical whether to print verbose output. Defaults to FALSE.
...	Additional arguments to pass on (not currently used)

Value

NULL Called for its side effect of printing a model summary to the console

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

See Also

Other Mplus-Formatting: [coef.mplus.model](#), [confint.mplus.model](#), [extract](#), [print.MplusRstructure](#)

Examples

```
## Not run:
# simple example of a model using builtin data
# demonstrates use
test <- mplusObject(
  TITLE = "test the MplusAutomation Package;",
  MODEL = "
    mpg ON wt hp;
    wt WITH hp;",
  usevariables = c("mpg", "wt", "hp"),
  rdata = mtcars)

res <- mplusModeler(test, "mtcars.dat", modelout = "model1.inp", run = 1L)

# example of the summary method
summary(res)

# example of verbose output
summary(res, verbose=TRUE)

# remove files
unlink("mtcars.dat")
unlink("model1.inp")
unlink("model1.out")
unlink("Mplus Run Models.log")

## End(Not run)
```

SummaryTable

Create a summary table of Mplus model statistics

Description

Creates output (optionally sent to a file) containing a summary table of model fit statistics extracted using the `extractModelSummaries` function. By default, the following summary statistics are included: Title, LL, Parameters, AIC, AICC, BIC, RMSEA_Estimate, but these are customizable using the `keepCols` and `dropCols` parameters.

Usage

```
SummaryTable(modelList, type = c("screen", "popup", "html", "latex",
  "markdown"), filename = "", keepCols, dropCols, sortBy, caption = "",
  display = FALSE, ..., include.rownames = FALSE)
```

Arguments

<code>modellist</code>	A list of models returned from the <code>extractModelSummaries</code> function.
<code>type</code>	A character vector indicating the type of output format to be generated. One of: “screen”, “popup”, “html”, “latex”, or “markdown”. Screen results in a simple summary table being sent to the R console.
<code>filename</code>	The name of the file to be created. Can be an absolute or relative path. If <code>filename</code> is a relative path or just the filename, then it is assumed that the file resides in the working directory <code>getwd()</code> . Example: “Mplus Summary.html”. By default, no filename is given, which results in the output being sent to the console. Note that currently, <code>filename</code> only has an effect for “html” and “latex”.
<code>keepCols</code>	A vector of character strings indicating which columns/variables to display in the summary. Only columns included in this list will be displayed (all others excluded). By default, <code>keepCols</code> is: <code>c("Title", "LL", "Parameters", "AIC", "AICC", "BIC", "RMSE")</code> . Example: <code>c("Title", "LL", "AIC", "CFI")</code>
<code>dropCols</code>	A vector of character strings indicating which columns/variables to omit from the summary. Any column not included in this list will be displayed. By default, <code>dropCols</code> is NULL. Example: <code>c("InputInstructions", "TLI")</code>
<code>sortBy</code>	optional. Field name (as character string) by which to sort the table. Typically an information criterion (e.g., “AIC” or “BIC”) is used to sort the table. Defaults to “AICC”.
<code>caption</code>	A character string, the caption to be given to the table. Currently only applies to types “html”, “latex”, and “markdown”.
<code>display</code>	optional logical (defaults to FALSE). This parameter specifies whether to display the table upon creation (TRUE or FALSE).
<code>include.rownames</code>	optional logical whether to include rownames or not.
<code>...</code>	additional arguments passed on to specific formatting types.

Value

Invisibly returns the summary table, which can be used if the printing options available are not sufficient.

Note

You must choose between `keepCols` and `dropCols` because it is not sensible to use these together to include and exclude columns. The function will error if you include both parameters.

Author(s)

Joshua F. Wiley based on code by Michael Hallquist

See Also

[extractModelSummaries](#)

Examples

```
## Not run:
m1 <- mplusObject(TITLE = "Reduced",
  MODEL = "mpg ON wt;", rdata = mtcars)
m1.fit <- mplusModeler(m1, "mtcars.dat", run = 1L)
m2 <- mplusObject(TITLE = "Full",
  MODEL = "mpg ON wt hp qsec;", rdata = mtcars)
m2.fit <- mplusModeler(m2, "mtcars.dat", run = 1L)

SummaryTable(list(m1.fit, m2.fit))
SummaryTable(list(m1.fit, m2.fit), type = "popup")
SummaryTable(list(m1.fit, m2.fit), type = "markdown",
  keepCols = c("Title", "Parameters", "LL", "AIC", "CFI", "SRMR"),
  caption = "Table of Model Fit Statistics",
  split.tables = 200)

# remove files
unlink("mtcars.dat")
unlink("mtcars.inp")
unlink("mtcars.out")
unlink("Mplus Run Models.log")

## End(Not run)
```

testBParamCompoundConstraint

Test inequality-constrained hypothesis for two or more parameters based on iterations of MCMC chains

Description

Tests an inequality-constrained hypothesis (van de Schoot, Hoijtink, Hallquist, & Boelen, in press) based on draws from the posterior distribution of the model parameters, which provides information about the proportion of the distribution that is in agreement with a given hypothesis. This function is used for more complex hypotheses about three or more parameters, whereas testBParamConstraint tests a simple two-parameter hypothesis.

Usage

```
testBParamCompoundConstraint(bparams, test)
```

Arguments

bparams	An object containing draws from the posterior distribution (class <code>mplus.model</code> or <code>mplus.bparameters</code>). Obtained by <code>SAVEDATA:BPARAMETERS</code> in <code>Mplus</code> and <code>getSavedata_Bparams</code> or <code>readModels</code> in <code>MplusAutomation</code> .
test	The R code defining the parameter test of three or more parameters. Example: <code>"(STAITOT.ON.CG > STAITOT.ON.UG) & (BDIM.ON.CG > BDIM.ON.UG)"</code> .

Details

This function accepts a `bparameters` object containing iterations of the MCMC chains (rows) for each model parameter (columns) and prints out the number and proportion of draws that are consistent with the requested hypothesis test.

The `test` argument is evaluated directly as R code, with the `bparams` object attached so that variable names are available directly in the environment. Because the goal is to evaluate the test for each draw from the posterior distribution, remember to use vector-based logic operators, not boolean operators. That is, stick to `&` or `|` for joining tests of parameters, rather than `&&` or `||` since the latter will return a single TRUE/FALSE, which is irrelevant.

An example test in R logic would be `"(STAITOT.ON.CG > STAITOT.ON.UG) & (BDIM.ON.CG > BDIM.ON.UG)"`.

Value

No value is returned by this function. Instead, two summary tables are printed to the screen containing the number and proportion of draws consistent with the hypothesis.

Author(s)

Michael Hallquist

See Also

[testBParamConstraint](#)

Examples

```
## Not run:
#using bparameters directly
btest <- getSavedata_Bparams("model vb1_simpel_b.out")
testBParametersCompoundConstraint(btest,
"(STDYX_STAITOT.ON.CG > STDYX_STAITOT.ON.UG) & (STDYX_BDIM.ON.CG > STDYX_BDIM.ON.UG)")

#or using readModels
btest <- readModels("model vb1_simpel_b.out")
testBParametersCompoundConstraint(btest,
"(STDYX_STAITOT.ON.CG > STDYX_STAITOT.ON.UG) & (STDYX_BDIM.ON.CG > STDYX_BDIM.ON.UG)")

## End(Not run)
```

testBParamConstraint	<i>Test inequality-constrained hypothesis for two parameters based on iterations of MCMC chains</i>
----------------------	---

Description

Tests a simple inequality-constrained hypothesis (van de Schoot, Hoijtink, Hallquist, & Boelen, in press) based on draws from the posterior distribution of the model parameters, which provides information about the proportion of the distribution that is in agreement with a given hypothesis. This function is used for simple hypothesis for two parameters, whereas testBParamCompoundConstraint gives full access to multiple parameters and R's logic syntax. This function accepts a bparameters object containing iterations of the MCMC chains (rows) for each model parameter (columns) and prints out the number and proportion of draws that are consistent with the requested hypothesis test. The coef1, operator, and coef2 arguments are appended in sequence, so that the hypothesis test is constructed from left-to-right. e.g., testBParamConstraint(bparamsDF, "MGM.TRT1", ">", "MGM.EX2").

Usage

```
testBParamConstraint(bparams, coef1, operator, coef2)
```

Arguments

bparams	An object containing draws from the posterior distribution (class <code>mplus.model</code> or <code>mplus.bparameters</code>). Obtained by <code>SAVEDATA:BPARAMETERS</code> in <code>Mplus</code> and <code>getSavedata_Bparams</code> or <code>readModels</code> in <code>MplusAutomation</code> .
coef1	The name of the first parameter to be compared. Example: "MGM.TRT1"
operator	A logical operator to compare the two parameters. Should be one of <code>>=</code> , <code>></code> , <code><</code> , or <code><=</code> . Example: " <code>>=</code> "
coef2	The name of the first parameter to be compared. Example: "MGM.EX2"

Value

No value is returned by this function. Instead, two summary tables are printed to the screen containing the number and proportion of draws consistent with the hypothesis.

Author(s)

Michael Hallquist

See Also

[testBParamCompoundConstraint](#)

Examples

```
## Not run:
#using bparameters directly
btest <- getSavedata_Bparams("model vb1_simpel_b.out")
testBParametersConstraint(btest, "STDYX_STAITOT.ON.CG", ">", "STDYX_STAITOT.ON.UCG")

#or using readModels
btest <- readModels("model vb1_simpel_b.out")
testBParametersConstraint(btest, "STDYX_STAITOT.ON.CG", ">", "STDYX_STAITOT.ON.UCG")

## End(Not run)
```

trainLGMM

Train a variety of latent growth mixture model

Description

This function iterates through a grid of values to train LGMMs, optionally using a local or remote cluster.

Usage

```
trainLGMM(data, idvar, assessmentvar, newdata = FALSE, tuneGrid, cl,
           ncores = 1L)
```

Arguments

data	A data frame or data table in long format (i.e., multiple rows per ID).
idvar	A character string of the variable name in the dataset that is the ID variable.
assessmentvar	A character string of the variable name in the dataset that indicates the particular assessment point for each timepoint.
newdata	A data frame of new values to use for generating predicted trajectories by class or FALSE if no predictions to be made (the default).
tuneGrid	A dataframe or list. It should have names for the needed arguments for long2LGMM().
cl	Optional. An existing cluster to be used to estimate models. Can be a local or remote cluster. In either case it needs MplusAutomation and Mplus available.
ncores	If a cluster is not passed to cl, specify the number of cores to use to create a local cluster. Must be an integer. Defaults to 1L.

Examples

```
## Not run:
## This example is not run by default because even with very limited number of
## random starts and iterations, it takes quite a few minutes
setwd(tempdir())

## Simulate Some Data from 3 classes
library(MASS)
set.seed(1234)
allcoef <- rbind(
  cbind(1, mvrnorm(n = 200,
                  mu = c(0, 2, 0),
                  Sigma = diag(c(.2, .1, .01)),
                  empirical = TRUE)),
  cbind(2, mvrnorm(n = 200,
                  mu = c(-3.35, 2, 2),
                  Sigma = diag(c(.2, .1, .1)),
                  empirical = TRUE)),
  cbind(3, mvrnorm(n = 200,
```

```

        mu = c(3.35, 2, -2),
        Sigma = diag(c(.2, .1, .1)),
        empirical = TRUE)))
allcoef <- as.data.frame(allcoef)
names(allcoef) <- c("Class", "I", "L", "Q")
allcoef$ID <- 1:nrow(allcoef)
d <- do.call(rbind, lapply(1:nrow(allcoef), function(i) {
  out <- data.frame(
    ID = allcoef$ID[i],
    Class = allcoef$Class[i],
    Assess = 1:11,
    x = sort(runif(n = 11, min = -2, max = 2)))
  out$y <- rnorm(11,
    mean = allcoef$I[i] + allcoef$L[i] * out$x + allcoef$Q[i] * out$x^2,
    sd = .1)
  return(out)
}))

## create splines
library(splines)
time_splines <- ns(d$x, df = 3, Boundary.knots = quantile(d$x, probs = c(.02, .98)))
d$t1 <- time_splines[, 1]
d$t2 <- time_splines[, 2]
d$t3 <- time_splines[, 3]
d$xq <- d$x^2

## create new data to be used for predictions
nd <- data.frame(ID = 1,
  x = seq(from = -2, to = 2, by = .1))
nd.splines <- with(attributes(time_splines),
  ns(nd$x, df = degree, knots = knots,
    Boundary.knots = Boundary.knots))
nd$t1 <- nd.splines[, 1]
nd$t2 <- nd.splines[, 2]
nd$t3 <- nd.splines[, 3]
nd$xq <- nd$x^2

## create a tuning grid of models to try
## all possible combinations are created of different time trends
## different covariance structures of the random effects
## and different number of classes
tuneGrid <- expand.grid(
  dv = "y",
  timevars = list(c("t1", "t2", "t3"), "x", c("x", "xq")),
  starts = "2 1",
  cov = c("independent", "zero"),
  k = c(1L, 3L),
  processors = 1L, run = TRUE,
  misstrick = TRUE, stringsAsFactors = FALSE)
tuneGrid$title <- paste0(
  c("linear", "quad", "spline")[sapply(tuneGrid$timevars, length)],
  "_",
  sapply(tuneGrid$cov, function(x) if(nchar(x)==4) substr(x, 1, 4) else substr(x, 1, 3)),

```

```

    "_",
    tuneGrid$k)
tuneGrid$base <- paste0(
  c("linear", "quad", "spline")[sapply(tuneGrid$timevars, length)],
  "_",
  sapply(tuneGrid$cov, function(x) if(nchar(x)==4) substr(x, 1, 4) else substr(x, 1, 3)))

## example using long2LGMM to fit one model at a time
mres <- long2LGMM(
  data = d,
  idvar = "ID",
  assessmentvar = "Assess",
  dv = tuneGrid$dv[1],
  timevars = tuneGrid$timevars[[1]],
  misstrick = tuneGrid$misstrick[1],
  k = tuneGrid$k[1],
  title = paste0(tuneGrid$title[1], tuneGrid$k[1]),
  base = tuneGrid$base[1],
  run = tuneGrid$run[1],
  processors = tuneGrid$processors[1],
  starts = tuneGrid$starts[1],
  newdata = nd,
  cov = tuneGrid$cov[1])

## Example using trainLGMM to fit a whole set of models
## can be distributed across a local or remote cluster
## Defaults to creating a local cluster, but can also pass an
## existing cluster
AllRes <- trainLGMM(
  data = d,
  idvar = "ID",
  assessmentvar = "Assess",
  newdata = nd,
  tuneGrid = tuneGrid,
  ncores = 2L)

tuneGridRes <- as.data.frame(
  cbind(tuneGrid,
    do.call(rbind, lapply(AllRes, function(x) {
      if (is.null(x$Model$results$summaries)) {
        NA
      } else {
        out <- x$Model$results$summaries
        ## deal with missing summary information for k = 1
        if (is.null(out$Entropy)) {
          out$Entropy <- 1
        }
        if (is.null(out$NCategoricalLatentVars)) {
          out$NCategoricalLatentVars <- 0
        }
        out[, sort(names(out)), drop = FALSE]
      }
    }
  )
)
```

```

    })))

tuneGridRes$Type <- gsub("[a-z]+_.*$", "\\1", tuneGridRes$title)

tuneGridRes$MinClass <- sapply(AllRes, function(x) {
  n <- x$Model$results$class_counts$mostLikely$count
  if(is.null(n)) {
    length(unique(d$ID))
  } else {
    min(n, na.rm = TRUE)
  }
})

## when trying many models, some may not converge
## subset to omit any missing AICC and look only at those with some
## minimum number of participants per class,
## for demonstration only arbitrarily set at 30
subset(tuneGridRes, !is.na(AICC) & MinClass >= 30,
       select = c(title, aBIC, AICC, Entropy, MinClass, LL))

## reshape data into long form which can make a very nice plot using ggplot2
tuneGridResL <- reshape(
  subset(tuneGridRes, select = c(Type, cov, k, Parameters, aBIC, AICC, AIC, BIC, Entropy)),
  varying = c("Parameters", "aBIC", "AICC", "AIC", "BIC", "Entropy"),
  v.names = "value",
  times = c("Parameters", "aBIC", "AICC", "AIC", "BIC", "Entropy"),
  timevar = "variable",
  idvar = c("Type", "cov", "k"),
  direction = "long")
tuneGridResL$cov <- factor(tuneGridResL$cov, levels = c("zero", "independent"))

## uncomment to run
## library(ggplot2)
## ggplot(tuneGridResL, aes(k, value, colour = Type, shape = Type)) +
##   geom_point() +
##   facet_grid(variable~cov, scales = "free")

## nice plot of the average trajectories in each class
## these are possible as trainLGMM exports predicted values for the
## new data fed in
## uncomment to run
## ggplot(AllRes[[which(tuneGridRes$title=="quad_ind_3")]]$predictions, aes(x)) +
##   geom_line(aes(y = y_1), colour = "black", size = 2) +
##   geom_line(aes(y = y_2), colour = "red", size = 2) +
##   geom_line(aes(y = y_3), colour = "blue", size = 2)

## End(Not run)

```

Description

This is a method for updating an Mplus model object. It takes an Mplus model object as the first argument, and then optionally any sections to update. There are two ways to update a section using a formula interface. `~ "new stuff"` will replace a given section with the new text. Alternately, you can add additional text using `~ + "additional stuff"`. Combined these let you replace or add to a section.

Usage

```
## S3 method for class 'mplusObject'
update(object, ...)
```

Arguments

<code>object</code>	An object of class <code>mplusObject</code>
<code>...</code>	Additional arguments to pass on

Value

An (updated) Mplus model object

Author(s)

Joshua F. Wiley <jwiley.psych@gmail.com>

Examples

```
example1 <- mplusObject(MODEL = "mpg ON wt;",
  usevariables = c("mpg", "hp"), rdata = mtcars)
x <- ~ "ESTIMATOR = ML;"
str(update(example1, rdata = iris))
str(update(example1, ANALYSIS = x))
str(update(example1, MODEL = ~ "wt ON hp;"))
str(update(example1, MODEL = ~ . + "wt ON hp;"))
str(update(example1, ANALYSIS = x, MODEL = ~ . + "wt ON hp;"))

# test to make sure . in Mplus code does not cause problems
str(update(example1, ANALYSIS = x, MODEL = ~ . + "wt ON hp*.5;"))
rm(example1, x)
```

Index

*Topic **datasets**

lcademo, [26](#)

*Topic **interface**

coef.mplus.model, [4](#)

compareModels, [5](#)

confint.mplus.model, [7](#)

createModels, [9](#)

createSyntax, [10](#)

detectVariables, [11](#)

extract, [12](#)

extractModelParameters, [13](#)

extractModelSummaries, [16](#)

extractModIndices, [19](#)

getSavedata_Bparams, [20](#)

getSavedata_Data, [21](#)

getSavedata_Fileinfo, [22](#)

HTMLSummaryTable, [23](#)

LatexSummaryTable, [25](#)

lookupTech1Parameter, [30](#)

mplus.traceplot, [31](#)

mplusObject, [39](#)

mplusRcov, [42](#)

parseCatOutput, [46](#)

plot.mplusObject, [48](#)

prepareMplusData, [49](#)

print.MplusRstructure, [52](#)

readModels, [53](#)

runModels, [56](#)

runModels_Interactive, [57](#)

separateHyphens, [58](#)

showSummaryTable, [59](#)

summary.mplusObject, [60](#)

SummaryTable, [61](#)

testBParamCompoundConstraint, [63](#)

testBParamConstraint, [64](#)

update.mplusObject, [69](#)

*Topic **package**

MplusAutomation, [32](#)

*Topic **utilities**

cd, [3](#)

*Topic **utils**

paramExtract, [45](#)

parseMplus, [47](#)

cd, [3](#)

coef.mplus.model, [4](#), [8](#), [12](#), [53](#), [61](#)

coef.mplusObject (coef.mplus.model), [4](#)

compareModels, [5](#), [33](#)

confint.mplus.model, [4](#), [7](#), [12](#), [53](#), [61](#)

confint.mplusObject

(confint.mplus.model), [7](#)

createModels, [9](#), [33](#)

createSyntax, [10](#)

detectVariables, [11](#)

extract, [4](#), [8](#), [12](#), [53](#), [61](#)

extract, mplus.model-method (extract), [12](#)

extract, mplusObject-method (extract), [12](#)

extract.mplus.model (extract), [12](#)

extract.mplusObject (extract), [12](#)

extractModelParameters, [13](#), [20](#), [33](#)

extractModelSummaries, [15](#), [16](#), [20](#), [24](#), [26](#),
[33](#), [60](#), [62](#)

extractModIndices, [19](#)

getSavedata_Bparams, [20](#), [63](#), [65](#)

getSavedata_Data, [21](#), [21](#), [23](#), [33](#)

getSavedata_Fileinfo, [21](#), [22](#), [22](#)

HTMLSummaryTable, [23](#), [26](#), [33](#), [60](#)

LatexSummaryTable, [24](#), [25](#), [33](#), [60](#)

lcademo, [26](#)

long2LGMM, [27](#)

lookupTech1Parameter, [30](#)

mplus.traceplot, [31](#)

MplusAutomation, [32](#)

MplusAutomation-package
 (MplusAutomation), 32
mplusModeler, 10–12, 33, 41, 48
mplusObject, 11, 39
mplusRcov, 42

paramExtract, 45
parseCatOutput, 46
parseMplus, 34, 47
plot.mcmc, 32
plot.mplusObject, 48
prepareMplusData, 10, 33, 34, 49
print.MplusRstructure, 4, 8, 12, 52, 61

readModels, 4, 8, 12, 18, 20, 30, 32, 33, 35,
 45, 46, 53, 63, 65
regex, 18
runModels, 18, 32, 33, 35, 56, 58
runModels_Interactive, 32, 57, 57

separateHyphens, 58
showSummaryTable, 24, 26, 33, 59
summary.mplusObject, 4, 8, 12, 53, 60
SummaryTable, 61
Sweave, 26

testBParamCompoundConstraint, 63, 65
testBParamConstraint, 64, 64
trainLGMM, 66

update.mplusObject, 69