Package 'MplusAutomation'

September 2, 2025

Type Package

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

Version 1.2

Date 2025-08-25

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 (<https://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://michaelhallquist.github.io/MplusAutomation/

BugReports https://github.com/michaelhallquist/MplusAutomation/issues

Depends R (>= 3.5.0), methods

Imports utils, boot, plyr, gsubfn, coda, xtable, lattice, texreg, pander, digest, parallel, ggplot2, data.table, fastDummies, checkmate

Suggests rhdf5, tcltk, relimp, knitr, testthat (>= 3.0.0), rmarkdown

LazyLoad yes

LazyData yes

VignetteBuilder knitr

RoxygenNote 7.3.2

Encoding UTF-8

Config/testthat/edition 3

NeedsCompilation no

Author Michael Hallquist [aut, cre],
Joshua Wiley [aut],
Caspar van Lissa [ctb],
Daniel Morillo [ctb]

Repository CRAN

Date/Publication 2025-09-02 20:50:47 UTC

Contents

`.mplusMultinomial` *Internal Function for Multinomial Regression in Mplus*

Description

Internal Function for Multinomial Regression in Mplus

Usage

```
.mplusMultinomial(
  dv,
  iv,
  data,
  idvar = "",
  integration = 1000,
  processors = 2,
  OR = TRUE,
  pairwise = TRUE,
  ...
)
```

Arguments

`dv`	A character string with the variable name for the dependent (outcome) variable.
`iv`	A character vector with the variable name(s) for the independent (predictor/explanatory) variable(s).
`data`	A dataset.
`idvar`	Optional. A character string indicating the name of the ID variable. Not currently used but may be used in future.
`integration`	An integer indicating the number of Monte Carlo integration points to use. Defaults to 1000.
`processors`	An integer indicating the number of processors to use. Passed to Mplus. Defaults to 2.
`OR`	A logical value whether odds ratios should be returned. Defaults to `TRUE`.
`pairwise`	A logical value indicating whether all pairwise tests should be computed. Defaults to `TRUE`.
`...`	Additional arguments passed to `mplusModeler()`.

Value

A list of results and Mplus model object.

Author(s)

Joshua F. Wiley jwiley.psych@gmail.com

Examples

```
## Not run:

set.seed(1234)
tmpd <- data.frame(
  x1 = rnorm(200),
  x2 = rnorm(200),
  x3 = cut(rnorm(200),
           breaks = c(-Inf, -.7, .7, Inf),
           labels = c("a", "b", "c")))
tmpd$y <- cut(rnorm(200, sd = 2) + tmpd$x1 + tmpd$x2 + I(tmpd$x3 == "b"),
              breaks = c(-Inf, -.5, 1, Inf),
              labels = c("L", "M", "H"))

tmpres <- MplusAutomation:::.mplusMultinomial(
  dv = "y",
  iv = c("x1", "x2"),
  data = tmpd,
  pairwise = TRUE)
tmpres2 <- MplusAutomation:::.mplusMultinomial(
  dv = "y",
  iv = c("x1", "x2"),
  data = tmpd,
  pairwise = FALSE)
tmpres3 <- MplusAutomation:::.mplusMultinomial(
  dv = "y",
  iv = c("x1@0", "x2@0"),
  data = tmpd,
  pairwise = FALSE)


## End(Not run)
```

---

cd *Change directory*

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)
```

checkSubmission *check on the status of submitted Mplus jobs on the cluster*

Description

check on the status of submitted Mplus jobs on the cluster

Usage

```
checkSubmission(mplus_submission_df = NULL, quiet = FALSE)
```

Arguments

mplus_submission_df	The data.frame returned by submitModels containing jobs to check on
quiet	If TRUE, do not print out the submission data.frame with current status

Value

invisibly, the mplus_submission_df with $status amd $status_time updated

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 mplud.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).

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`.
`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

Model outputs to be compared should come from the readModels command.

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

- "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": Print 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., <= 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:

- "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

A list containing the results of the comparison. Elements include summaries with selected summary statistics for both models, parameters listing equal and differing parameters as well as those unique to each model, and, when diffTest = TRUE, a diffTest element with the chi-square difference test.

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

`object`	An object of class mplusObject
`parm`	Included as all `confint()` methods must include it. Not used currently for Mplus.
`level`	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)
```

`createMixtures` *Create syntax for a batch of mixture models*

Description

Dynamically creates syntax for a batch of mixture models, with intelligent defaults. This function is a wrapper around `mplusObject` and `mplusModeler`, and the respective arguments of those functions can be passed on using `...`. For instance, passing the argument `run = 1L` means that the models will be evaluated and returned.

Usage

```
createMixtures(
  classes = 1L,
  filename_stem = NULL,
  model_overall = NULL,
  model_class_specific = NULL,
  rdata = NULL,
  usevariables = NULL,
  OUTPUT = "TECH11 TECH14;",
  SAVEDATA = "FILE IS {filename_stem}_{C}.dat;  SAVE = cprobabilities;",
  quiet = TRUE,
  ...
)
```

Arguments

`classes`	A vector of integers, indicating which class solutions to generate. Defaults to 1L. E.g., `classes = 1:6`, `classes = c(1:4, 6:8)`.
`filename_stem`	Character. A stem for the automatically generated filenames of the syntax and data files.
`model_overall`	Character. Mplus syntax for the overall model (across classes).
`model_class_specific`	Character vector. Mplus syntax for the class-specific model(s) of one or more categorical latent variables. Each element of `model_class_specific` is used as the class-specific syntax of a different categorical latent variable. This allows one to easily specify latent transition analyses (see second example). The character string "{C}" is substituted with the correct class number, for example to set unique parameter labels for each class, or to specify equality constraints.
`rdata`	Data.frame. An R dataset to be used for the model.
`usevariables`	Character vector, specifying the names of variables in the rdata object which should be included in the Mplus data file and model.
`OUTPUT`	Character. Syntax for Mplus' OUTPUT option. Highly recommended when determining the appropriate number of latent classes. TECH11 is required to obtain the VLMR-test; TECH14 is required for the BLR-test.

SAVEDATA	Character. Syntax for Mplus' savedata option. Highly recommended when conducting mixture models. The default option will often be adequate.
quiet	optional. If TRUE, show status messages in the console.
...	Additional arguments, passed to mplusObject, such as syntax for other Mplus options.

Details

In the arguments model_class_specific and SAVEDATA, the character string "{C}" is substituted with the correct class number. The character string "{filename_stem}" is substituted with the filename stem, for example, to name savedata in line with the input files.

In all arguments to mplusObject, a double space (" ") is replaced with a newline character. This can be used to obtain nicely formatted Mplus syntax.

Value

None, unless the argument run = 1L is specified. In that case, a list with elements of class mplusObject is returned. Otherwise, this function is used for its side effects (generating syntax).

Author(s)

Caspar J. van Lissa

See Also

mplusObject, mplusModeler

Examples

```
## Not run:
createMixtures(classes = 1:3, filename_stem = "iris", rdata = iris)

## End(Not run)
## Not run:
mydat <- read.csv(
system.file("extdata", "ex8.13.csv", package = "MplusAutomation"))
createMixtures(
classes = 2,
filename_stem = "dating",
model_overall = "c2 ON c1;",
model_class_specific = c(
"[u11$1] (a{C});  [u12$1] (b{C});  [u13$1] (c{C});  [u14$1] (d{C});  [u15$1] (e{C});",
"[u21$1] (a{C});  [u22$1] (b{C});  [u23$1] (c{C});  [u24$1] (d{C});  [u25$1] (e{C});"
),
rdata = mydat,
ANALYSIS = "PROCESSORS IS 2;  LRTSTARTS (0 0 40 20);  PARAMETERIZATION = PROBABILITY;",
VARIABLE = "CATEGORICAL = u11-u15 u21-u25;"
)

## 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

object	An object of class mplusObject
filename	The name of the data file as a character vector
check	A logical indicating whether or not to run parseMplus on the created input file. Checks for errors like lines that are too long, or for missing semi-colons and gives notes.
add	A logical passed on to parseMplus whether to add semi colons to line ends. Defaults to FALSE.
imputed	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"), file=stdout(), 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"),
  file=stdout(),
  fill=TRUE)
rm(example1)
```

---

detectMplus *Detect the location/name of the Mplus command*

Description

This is an utility function to help detect the correct name/path to Mplus. It tries hard across operating systems to find Mplus and if it cannot find the full version of Mplus to find a demo version of Mplus.

Usage

```
detectMplus()
```

Details

It does not require any arguments.

Value

A character string that is the Mplus command possibly the path to the mplus command or an error if it cannot be found.

Author(s)

Joshua F. Wiley jwiley.psych@gmail.com

Examples

```
## if you have Mplus installed, uncomment and run
## this will give an error if it cannot find Mplus.
## detectMplus()
```

---

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

model	An Mplus model object. This typically comes either from readModels directly, or indirectly via mplusModeler. The results will have different classes, but extract methods are defined for both.
summaries	A character vector which summaries to include. Defaults to "none".
cis	A logical whether to extract confidence intervals.
escape.latex	A logical value whether to escape dollar signs in coefficient names for LaTeX. Defaults to FALSE.
...	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)
```

`get_results` *Extract Mplus results*

Description

This function allows users to extract elements of Mplus output by name from different types of objects returned by `MplusAutomation`.

Usage

```
get_results(x, element, simplify = FALSE, ...)

get_input(x, simplify = FALSE, ...)

get_warn_err(x, simplify = FALSE, ...)

get_data_summary(x, simplify = FALSE, ...)

get_sampstat(x, simplify = FALSE, ...)

get_covariance_coverage(x, simplify = FALSE, ...)

get_summaries(x, simplify = FALSE, ...)

get_invariance_testing(x, simplify = FALSE, ...)

get_parameters(x, simplify = FALSE, ...)

get_class_counts(x, simplify = FALSE, ...)

get_indirect(x, simplify = FALSE, ...)

get_mod_indices(x, simplify = FALSE, ...)
```

```
get_residuals(x, simplify = FALSE, ...)

get_savedata(x, simplify = FALSE, ...)

get_bparameters(x, simplify = FALSE, ...)

get_tech1(x, simplify = FALSE, ...)

get_tech3(x, simplify = FALSE, ...)

get_tech4(x, simplify = FALSE, ...)

get_tech7(x, simplify = FALSE, ...)

get_tech8(x, simplify = FALSE, ...)

get_tech9(x, simplify = FALSE, ...)

get_tech10(x, simplify = FALSE, ...)

get_tech12(x, simplify = FALSE, ...)

get_tech15(x, simplify = FALSE, ...)

get_fac_score_stats(x, simplify = FALSE, ...)

get_lcCondMeans(x, simplify = FALSE, ...)

get_gh5(x, simplify = FALSE, ...)
```

Arguments

x	Object from which to extract results.
element	Which element of the results to extract.
simplify	Logical; should the result be simplified to a vector, matrix or higher dimensional array if possible? See sapply. Defaults to FALSE.
...	Additional arguments passed to and from functions.

Value

An atomic vector or matrix or list of the same length as X (of length n for replicate). If simplification occurs, the output type is determined from the highest type of the return values in the hierarchy NULL < raw < logical < integer < double < complex < character < list < expression, after coercion of pairlists to lists.

Examples

```
## Not run:
```

```
 test <- mplusObject(MODEL = "mpg ON wt hp;
 wt WITH hp;", rdata = mtcars)
 res <- mplusModeler(test, modelout = "model1.inp", run = 1L)
 get_results(res, "summaries")
 unlink(res$results$input$data$file)
 unlink("model1.inp")
 unlink("model1.out")

## End(Not run)
out <- get_input(res)
out <- get_warn_err(res)
out <- get_data_summary(res)
out <- get_sampstat(res)
out <- get_covariance_coverage(res)
out <- get_summaries(res)
out <- get_invariance_testing(res)
out <- get_parameters(res)
out <- get_class_counts(res)
out <- get_indirect(res)
out <- get_mod_indices(res)
out <- get_residuals(res)
out <- get_savedata(res)
out <- get_bparameters(res)
out <- get_tech1(res)
out <- get_tech3(res)
out <- get_tech4(res)
out <- get_tech7(res)
out <- get_tech8(res)
out <- get_tech9(res)
out <- get_tech10(res)
out <- get_tech12(res)
out <- get_tech15(res)
out <- get_fac_score_stats(res)
out <- get_lcCondMeans(res)
out <- get_gh5(res)
```

`HTMLSummaryTable` *Create an HTML file containing a summary table of Mplus model statistics*

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_Estimate`, but these are customizable using the `keepCols` and `dropCols` parameters.

Usage

```
HTMLSummaryTable(
```

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

Arguments

modelList	A list of models (as a data.frame) returned from the extractModelSummaries function.
filename	The name of the HTML file to be created. Can be an absolute or relative path. If filename is a relative path or just the filename, then it is assumed that the file resides in the working directory getwd(). Example: "Mplus Summary.html"
keepCols	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, keepCols is: c("Title", "LL", "Parameters", "AIC", "AICC", "BIC", "RMSEA_Estimate"). Example: c("Title", "LL", "AIC", "CFI")
dropCols	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, dropCols is NULL. Example: c("InputInstructions", "TLI")
sortBy	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 NULL, which does not sort the table.
display	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` *Display summary table of Mplus model statistics in separate window*

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`, `RMSEA_Estimate`, but these are customizable using the `keepCols` and `dropCols` parameters.

Usage

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

Arguments

`modelList`	A list of models (as a `data.frame`) returned from the `extractModelSummaries` function.
`keepCols`	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, `keepCols` is: `c("Title", "LL", "Parameters", "AIC", "AICC", "BIC", "RMSEA_Estimate")`. Example: `c("Title", "LL", "AIC", "CFI")`
`dropCols`	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, `dropCols` is `NULL`. Example: `c("InputInstructions", "TLI")`
`sortBy`	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 `NULL`, which does not sort the table.
`label`	optional. A character string specifying the label for the LaTex table, which can be used for referencing the table.
`caption`	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 ismilar 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 mplusModeler().

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 to 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)
```

`mixtureSummaryTable` *Create a summary table of Mplus mixture models*

Description

Creates a summary table of model fit statistics and relevant diagnostic information for a list of mixture models. Default statistics reported are in line with published guidelines (see Jung & Wickrama, 2008; Nylund et al., 2007): `c("Title", "Classes", "Warnings", "AIC", "BIC", "aBIC","Entropy", "T11_VLMR_PValue", "T11_LMR_PValue", "BLRT_PValue", "min_N","max_N", "min_prob", "max_prob")`. The table is customizable using the `keepCols` parameter, which is passed through to SummaryTable.

Usage

```
mixtureSummaryTable(
  modelList,
  keepCols = c("Title", "Classes", "Warnings", "AIC", "BIC", "aBIC", "Entropy",
   "T11_VLMR_PValue", "T11_LMR_PValue", "BLRT_PValue", "min_N", "max_N", "min_prob",
     "max_prob"),
  sortBy = NULL,
  ...
)
```

Arguments

`modelList`	A list of models returned from the `extractModelSummaries` function.
`keepCols`	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, `keepCols` is: `c("Title", "Classes", "Warnings", "AIC", "BIC", "aBIC","Entropy","T11_VLMR_PValue", "T11_LMR_PValue", "BLRT_PValue", "min_N", "max_N","min_prob", "max_prob")`.
`sortBy`	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". Set to NULL by default, so the table is ordered by increasing number of classes.
`...`	Arguments passed to `SummaryTable`.

Value

An object of class data.frame.

Note

This function is partially a wrapper around SummaryTable, with enhancements for summarizing mixture models.

Author(s)

Caspar J. van Lissa

See Also

SummaryTable

Examples

```
## Not run:
res <- createMixtures(classes = 1:2, filename_stem = "iris", rdata = iris,
               OUTPUT = "tech11 tech14;",
               run = 1L)
mixtureSummaryTable(res)

## End(Not run)
```

---

mplus.traceplot *Plot the samples for each MCMC chain as a function of iterations*

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:

- To automatically run groups/batches of models.
- To provide routines to extract model fit statistics, parameter estimates, and raw data from 'Mplus' output files.
- To facilitate comparisons among models
- 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).

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:	1.2
Date:	2025-08-25
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.

`mplusAvailable` *Check whether Mplus can be found*

Description

This is a simple utility to check whether Mplus can be found. Returns 0 if Mplus command can be found by the system. If `silent = FALSE`, prints a message to the user to help suggest what to do.

Usage

```
mplusAvailable(silent = TRUE)
```

Arguments

`silent`	A logical whether to print a message or not. Defaults to `TRUE` for silent operation.

Value

The status of finding Mplus. Per unix conventions, status 0 indicates Mplus was found (0 problems) and status 1 indicates that Mplus was not found.

Author(s)

Joshua Wiley

Examples

```
mplusAvailable(silent = TRUE)
mplusAvailable(silent = FALSE)
```

mplusGLM *Function to fit GLMs in Mplus*

Description

The purpose of this function is to make it (relatively) easy to fit (most) generalized linear models in Mplus. Fitting GLMs in Mplus offers advantages such as using full information maximum likelihood for missing data, robust estimators (default used is MLR), and standard errors adjusted for clustering (planned; not currently available via `mplusGLM()`. The overarching aim of this function is to make most GLMs as easy to fit in Mplus as they are in R.

Usage

```
mplusGLM(formula, data, idvar = "", ...)
```

Arguments

`formula`	An R formula class object as used in `glm()`. Note that currently, only basic formula are accepted. On the fly recoding, arthimetic, and on the fly interactions do not currently work.
`data`	A dataset.
`idvar`	Optional. A character string indicating the name of the ID variable. Not currently used but may be used in future.
`...`	Additional arguments passed to helper functions. For example `.mplusMultinomial()`.

Details

Note that although there are benefits to fitting GLMs in Mplus. Caution also is warranted. Using full information maximum likelihood for missing data requires a number of assumptions. These may be (badly) violated. `mplusGLM()` requires the analyst to check these as appropriate.

Currently, `mplusGLM()` only supports multinomial outcomes. More outcomes are planned in the future including binary, continuous/normal, and count outcomes.

Value

A list of results and Mplus model object.

Author(s)

Joshua F. Wiley jwiley.psych@gmail.com

Examples

```
## Not run:
set.seed(1234)
tmpd <- data.frame(
  x1 = rnorm(200),
  x2 = rnorm(200),
  x3 = cut(rnorm(200),
           breaks = c(-Inf, -.7, .7, Inf),
           labels = c("a", "b", "c")))
tmpd$y <- cut(rnorm(200, sd = 2) + tmpd$x1 + tmpd$x2 + I(tmpd$x3 == "b"),
              breaks = c(-Inf, -.5, 1, Inf),
              labels = c("L", "M", "H"))
test <- mplusGLM(y ~ x1 + x2 + x3, data = tmpd)

## End(Not run)
```

---

mplusModel *Create an mplusModel object for a given model*

Description

Create an mplusModel object for a given model

Usage

```
mplusModel(
  syntax = NULL,
  data = NULL,
  inp_file = NULL,
  read = TRUE,
  Mplus_command = NULL
)
```

Arguments

`syntax`	a character vector of Mplus input syntax for this model
`data`	a data.frame to be used for estimating the model
`inp_file`	the location of .inp file for this model
`read`	If TRUE and the .out file already exists, read the contents of the .out file using `readModels`
`Mplus_command`	The location of the Mplus executable to run. If NULL, use `detectMplus()`

Value

a `mplusModel_r6` object containing information about the 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 = detectMplus(),
  writeData = c("ifmissing", "always", "never"),
  hashfilename = TRUE,
  killOnFail = TRUE,
  quiet = TRUE,
  ...
)
```

Arguments

object	An object of class mplusObject
dataout	the name of the file to output the data to for Mplus. If missing, defaults to modelout changing .inp to .dat.
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 .inp. If missing, defaults to dataout changing the extension to .inp.
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 run replications as well as the basic model.
check	logical whether the body of the Mplus syntax should be checked for missing semicolons using the parseMplus function. Defaults to FALSE.
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 prepareMplusData. Note that previously, mplusModeler 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 TRUE in mplusModeler. Note that this behavior is a change from previous versions and differs from prepareMplusData which maintains the old behavior by default of FALSE.
killOnFail	A logical whether or not to kill any mplus processes on failure. Passed on to control behavior of runModels. Defaults to TRUE.
quiet	optional. If TRUE, show status messages in the console.
...	additional arguments passed to the prepareMplusData 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 tedius 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 mroe 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 (l1-l5);
  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))




# can write multiply imputed data too
# here are three "imputed" datasets
idat <- list(
  data.frame(mpg = mtcars$mpg, hp = c(100, mtcars$hp[-1])),
  data.frame(mpg = mtcars$mpg, hp = c(110, mtcars$hp[-1])),
  data.frame(mpg = mtcars$mpg, hp = c(120, mtcars$hp[-1])))

# if we turn on hashing in the filename the first time,
# we can avoid overwriting notes the second time
testobjimp <- mplusObject(MODEL = "[mpg];", rdata = idat, imputed = TRUE)

testimp <- mplusModeler(
  testobjimp,
  modelout = "testimp.inp",
  writeData = "ifmissing", hashfilename=FALSE)

testimp <- mplusModeler(
  testobjimp,
  modelout = "testimp.inp",
  writeData = "ifmissing", hashfilename=TRUE)

testimp <- mplusModeler(
  testobjimp,
  modelout = "testimp.inp",
  writeData = "ifmissing", hashfilename=TRUE,
  run = TRUE)
```

```
testobjimp2 <- mplusObject(MODEL = "[hp];", rdata = idat, imputed = TRUE)
testimp2 <- mplusModeler(
  testobjimp2,
  modelout = "testimp2.inp",
  writeData = "ifmissing", hashfilename=TRUE,
  run = TRUE)

 # 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,
```

```
  quiet = TRUE,
  ...
)
```

Arguments

TITLE	A character string of the title for Mplus.
DATA	A charater 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, autov is 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.
quiet	optional. If TRUE, show status messages in the console.
...	Arguments passed on to mplusModeler if run > 0.

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 = "basename_01 ON basename_02;",
  rdata = data.frame(basename_01 = 1:5, basename_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:

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

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

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 `readModels`) 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 `|`). 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), file=stdout(), fill=TRUE)
# add missing semicolons and return
cat(parseMplus(test, TRUE), file=stdout(), fill=TRUE)
# line that is too long for Mplus
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), file=stdout())
```

---

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)
```

---

plotMixtureDensities *Create density plots for mixture models*

Description

Creates a density plot for a single object of class 'mplus.model', or a faceted plot of density plots for an object of class 'mplus.model.list'. For each variable, a Total density plot will be shown, along with separate density plots for each latent class, where cases are weighted by the posterior probability of being assigned to that class.

Usage

```
plotMixtureDensities(
  modelList,
  variables = NULL,
```

```
  bw = FALSE,
  conditional = FALSE,
  alpha = 0.2,
  facet_labels = NULL
)
```

Arguments

modelList	A list object of Mplus models, or a single Mplus model
variables	Which variables to plot. If NULL, plots all variables that are present in all Mplus models.
bw	Logical. Whether to make a black and white plot (for print) or a color plot. Defaults to FALSE, because these density plots are hard to read in black and white.
conditional	Logical. Whether to show a conditional density plot (surface area is divided amongst the latent classes), or a classic density plot (surface area of the total density plot is equal to one, and is subdivided amongst the classes).
alpha	Numeric (0-1). Only used when bw and conditional are FALSE. Sets the transparency of geom_density, so that classes with a small number of cases remain visible.
facet_labels	Named character vector, the names of which should correspond to the facet labels one wishes to rename, and the values of which provide new names for these facets. For example, to rename variables, in the example with the 'iris' data below, one could specify: `facet_labels = c("Pet_leng" = "Petal length")`.

Value

An object of class 'ggplot'.

Note

This function returns warnings, indicating that sum(weights) != 1. These can be ignored. The sum of the "Total" density per variable per model is equal to 1, and the sum of all of the posterior probabilities is equal to

1. This results in a normal density plot for the "Total", which is subdivided by the latent classes, in proportion to the posterior probabilities of participants being assigned to those clases.

Author(s)

Caspar J. van Lissa

Examples

```
## Not run:
results <- createMixtures(classes = 1:3, filename_stem = "iris",
                          rdata = iris, run = 1L)
plotMixtureDensities(results)
```

```
## End(Not run)
## Not run:
plotMixtureDensities(results, variables = "PETAL_LE")

## End(Not run)
## Not run:
plotMixtureDensities(results, bw = TRUE)

## End(Not run)
## Not run:
plotMixtureDensities(results, bw = FALSE, conditional = TRUE)

## End(Not run)
## Not run:
plotMixtureDensities(results[[2]], variables = "PETAL_LE")

## End(Not run)
```

`plotMixtures` *Create latent profile plots*

Description

Creates a profile plot for a single object of class 'mplus.model', or a faceted plot of profile plots for an object of class 'mplus.model.list'.

Usage

```
plotMixtures(
  modelList,
  variables = NULL,
  coefficients = c("unstandardized", "stdyx.standardized", "stdy.standardized",
    "stdy.standardized"),
  parameter = c("Means", "Intercepts"),
  ci = 0.95,
  bw = FALSE,
  rawdata = FALSE,
  alpha_range = c(0, 0.1)
)
```

Arguments

`modelList`	A list of Mplus mixture models, or a single mixture model
`variables`	A character vectors with the names of the variables (included in the Mplus output) to be plotted.
`coefficients`	Which type of coefficients to plot on the y-axis; default is 'unstandardized'. Options include: `c('stdyx.standardized', 'stdy.standardized', 'std.standardized')`

parameter	Which parameter to plot (from Mplus parameter estimate headings included in the output). Defaults to `c('Means', 'Intercepts')`.
ci	What confidence interval should the errorbars span? Defaults to a 95% confidence interval. Set to NULL to remove errorbars.
bw	Logical. Should the plot be black and white (for print), or color?
rawdata	Should raw data be plotted in the background? Setting this to TRUE might result in long plotting times. Requires including the Mplus syntax 'SAVEDATA: FILE IS "filename"; SAVE = cprobabilities' in the Mplus input.
alpha_range	The minimum and maximum values of alpha (transparancy) for the raw data. Minimum should be 0; lower maximum values of alpha can help reduce overplotting.

Value

An object of class 'ggplot'.

Author(s)

Caspar J. van Lissa

Examples

```
## Not run:
res <- createMixtures(classes = 1:2, filename_stem = "cars",
                      model_overall = "wt ON drat;",
                      model_class_specific = "wt;  qsec;",
                      rdata = mtcars,
                      usevariables = c("wt", "qsec", "drat"),
                      OUTPUT = "standardized",
                      run = 1L)
plotMixtures(res, rawdata = TRUE)

## End(Not run)
## Not run:
plotMixtures(res, variables = "wt")

## End(Not run)
## Not run:
plotMixtures(res, coefficients = "stdyx.standardized")

## End(Not run)
```

`prepareMplusData` *Create tab-delimited file and Mplus input syntax from R data.frame*

Description

The prepareMplusData function converts an R data.frame (or a list of data frames), 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 = NULL,
  inpfile = FALSE,
  keepCols = NULL,
  dropCols = NULL,
  dummyCode = NULL,
  interactive = TRUE,
  overwrite = TRUE,
  imputed = FALSE,
  writeData = c("always", "ifmissing", "never"),
  hashfilename = FALSE,
  quiet = TRUE,
  use_relative_path = 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"
inpfile	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.
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.
dummyCode	An optional character vector of column names indicating categorical variables in the dataset that should be converted into dummy codes (using the fastDummies package). Note that one dummy code is returned for *each level*, so no reference category is implied. Thus, it is up to you to drop one of the dummy codes in the Mplus syntax to denote the reference category and avoid multicollinearity.
interactive	Logical value indicating whether file names should be selected interactively. If filename is missing and interative=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.
quiet	optional. If TRUE, show status messages in the console.
use_relative_path	If TRUE, only include the relative path in the DATA: FILE = syntax returned by the function. This works well if the .dat file and the .inp file are located in the same folder, as is common for Mplus. Default: FALSE.

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

# example when there is a factor and logical
tmpd <- mtcars
tmpd$cyl <- factor(tmpd$cyl)
tmpd$am <- as.logical(tmpd$am)
prepareMplusData(tmpd, "test_type.dat")
rm(tmpd)

# 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)

# now that the filename was hashed in test01c, future calls do not re-write data
# as long as the hash matches
test01db <- prepareMplusData(mtcars, "test01d.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", inpfile=TRUE)

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

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


# can write multiply imputed data too
# here are three "imputed" datasets
idat <- list(
  data.frame(mpg = mtcars$mpg, hp = c(100, mtcars$hp[-1])),
  data.frame(mpg = mtcars$mpg, hp = c(110, mtcars$hp[-1])),
  data.frame(mpg = mtcars$mpg, hp = c(120, mtcars$hp[-1])))

# if we turn on hashing in the filename the first time,
# we can avoid overwriting notes the second time
testimp1 <- prepareMplusData(idat, "testi1.dat",
  writeData = "ifmissing", hashfilename=TRUE,
  imputed = TRUE)

# now that the filename was hashed, future calls do not re-write data
# as long as all the hashes match
testimp2 <- prepareMplusData(idat, "testi2.dat",
  writeData = "ifmissing", hashfilename=TRUE,
  imputed = TRUE)

# in fact, the number of imputations can decrease
# and they still will not be re-written
testimp3 <- prepareMplusData(idat[-3], "testi3.dat",
  writeData = "ifmissing", hashfilename=TRUE,
  imputed = TRUE)

# however, if the data changes, then all are re-written
# note that it warns for the two files that already exist
# as these two are overwritten
```

```
idat2 <- list(
  data.frame(mpg = mtcars$mpg, hp = c(100, mtcars$hp[-1])),
  data.frame(mpg = mtcars$mpg, hp = c(109, mtcars$hp[-1])),
  data.frame(mpg = mtcars$mpg, hp = c(120, mtcars$hp[-1])))
testimp4 <- prepareMplusData(idat2, "testi4.dat",
  writeData = "ifmissing", hashfilename=TRUE,
  imputed = TRUE)



## 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 *Read Parameters, Summary Statistics, and Savedata from Mplus Output*

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,
  pathfilter,
  what = "all",
  quiet = TRUE
)
```

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 based on their file name. See regex or https://www.pcre.org/pcre.txt for details about regular expression syntax.
pathfilter	a Perl regular expression (PCRE-compatible) specifying particular paths to be parsed within directory based on their absolute paths.
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 TRUE.

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", "data_summary", "sampstat", "covariance_coverage", "summaries", "parameters", "class_counts", "indirect", "mod_indices", "residuals", "savedata", "bparameters", "tech1", "tech3", "tech4", "tech7", "tech8", "tech9", "tech10", "tech12", "fac_score_stats", "lcCondMeans", "gh5", "output")

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
- `data_summary`: Output of SUMMARY OF DATA section, including cluster sizes and ICCs
- `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 each indirect effect section
- `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
- `tech10`: a list containing model fit information from OUTPUT: TECH10

- 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
- r3step: predictors of latent class membership using the 3-step procedure (R3STEP)
- gh5: a list containing data from the gh5 (graphics) file corresponding to this output. (Requires rhdf5 package)
- h5results: a list containing data from h5results file produced by Mplus v8.11+. (Requires rhdf5 package)
- output: The entire, raw output file.

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 = detectMplus(),
  killOnFail = TRUE,
  local_tmpdir = FALSE,
  quiet = TRUE
)
```

Arguments

target	a character vector where each element is a directory containing Mplus input files (.inp) to run OR a single .inp file to be run. Elements 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 among those found in target. See regex or https://www.pcre.org/pcre.txt for details about regular expression syntax.
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.
quiet	optional. If FALSE, show status messages in the console.

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)
## Not run:
  runModels(getwd(), filefilter = "ex8.*", logFile=NULL)

## End(Not run)
```

---

runModels_Interactive *Run Mplus Models Using Graphical Interface*

Description

This function is 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` *Separate Hyphenated Variable Strings*

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 *Display summary table of Mplus model statistics in separate window*

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 = NULL,
  font = "Courier 9"
)
```

Arguments

`modelList`	A list of models (as a `data.frame`) returned from the `extractModelSummaries` function.
`keepCols`	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, `keepCols` is: `c("Title", "LL", "Parameters", "AIC", "AICC", "BIC", "RMSEA_Estimate")`. Example: `c("Title", "LL", "AIC", "CFI")`
`dropCols`	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, `dropCols` is `NULL`. Example: `c("InputInstructions", "TLI")`
`sortBy`	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 `NULL`, which does not sort the table.
`font`	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!!!
```

`submitModels` *Submit Mplus models to a high-performance cluster scheduler*

Description

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

Usage

```
submitModels(
  target = getwd(),
  recursive = FALSE,
  filefilter = NULL,
  replaceOutfile = "modifiedDate",
  Mplus_command = NULL,
  quiet = FALSE,
  scheduler = "slurm",
  sched_args = NULL,
  env_variables = NULL,
  export_all = FALSE,
  cores_per_model = 1L,
  memgb_per_model = 8L,
  time_per_model = "1:00:00",
  pre = NULL,
  post = NULL,
```

```
  batch_outdir = NULL,
  job_script_prefix = NULL,
  combine_jobs = TRUE,
  max_time_per_job = "24:00:00",
  combine_memgb_tolerance = 1,
  combine_cores_tolerance = 2,
  debug = FALSE,
  fail_on_error = TRUE
)
```

Arguments

target	a character vector where each element is a directory containing Mplus input files (.inp) to run OR a single .inp file to be run. Elements 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 among those found in target. See regex or https://www.pcre.org/pcre.txt for details about regular expression syntax.
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).
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.
quiet	optional. If FALSE, show status messages in the console.
scheduler	Which scheduler to use for job submission. Options are 'qsub', 'torque', 'sbatch', 'slurm', 'local', or 'sh'. The terms 'qsub' and 'torque' are aliases (where 'torque' submits via the qsub command). Likewise for 'sbatch' and 'slurm'. If 'local' or 'sh' are specified, submitModels does not submit to any scheduler at all, but instead executes the command locally via a shell script.
sched_args	A character vector of arguments to be included in the scheduling command. On TORQUE, these will typically begin with '-l' such as '-l wall_time=10:00:00'. These are added inside the submission script for each model and are shared across all models. To add model-specific arguments, include ! #SBATCH or ! #PBS lines inside the individual .inp files
env_variables	A named character vector containing environment variables and their values to be passed to the script at execution time. This is handled by the -v directive on TORQUE clusters and by --export on Slurm clusters. The names of this vector are the environment variable names and the values of the vector are the

environment variable values to be passed in. If you want to propagate the current value of an environment variable to the compute node at runtime, use NA as the value of the element in `env_variables`. See examples.

`export_all` Whether to export all environment variables to the compute node at runtime. Default: FALSE

`cores_per_model`
How many cpus/cores are requested for each model (can be overriden using `! BATCH` directives in .inp files). Default: 1.

`memgb_per_model`
amount of memory (RAM) requested for each model (in GB). Default: 8.

`time_per_model` amount of time requested for each model. Default: "1:00:00" (1 hour). If a number is provided, we will treat this as the number of minutes.

`pre` user-specified shell commands to include in the job script prior to running Mplus (e.g., module load commands)

`post` user-specified shell commands to include in the job script after Mplus runs (e.g., execute results wrangling script)

`batch_outdir` the directory where job scripts should be written

`job_script_prefix`
the filename prefix for each job script

`combine_jobs` if TRUE, `submitModels` will seek to combine similar models into batches to reduce the total number of jobs

`max_time_per_job`
The maximum time (in days-hours:minutes:seconds format) allowed for a combined job

`combine_memgb_tolerance`
The memory tolerance for combining similar models in GB. Defaults to 1 (i.e., models that differ by <= 1 GB can be combined)

`combine_cores_tolerance`
The cores tolerance for combining models with similar core requests. Defaults to 2 (i.e., models whose core requests differ by <= 2 can be combined)

`debug` a logical indicating whether to actually submit the jobs (TRUE) or just create the scripts for inspection (FALSE)

`fail_on_error` Whether to stop execution of the script (TRUE), or issue a warning (FALSE) if the job submission fails. Defaults to TRUE.

Details

Note that if `fail_on_error` is `TRUE` and submission of one model fails, the submission loop will stop, rather than submitting further models.

Value

A data.frame recording details of the jobs submitted by `submitModels`. This can be passed to the `summary` function or to `checkSubmission` to see the state of submitted jobs.

Author(s)

Michael Hallquist

Examples

```
## Not run:
 submitModels("~/Michael/submitTest", recursive=TRUE, sched_args=c("--mail=user", "--export=v"),
    max_time_per_job = "2:10:00", combine_jobs = TRUE)

## End(Not run)
```

---

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)
```

`summary.mplus_submission_df` *summary function for submission from* `submitModels`

Description

summary function for submission from `submitModels`

Usage

```
## S3 method for class 'mplus_submission_df'
summary(object, refresh = TRUE, ...)
```

Arguments

`object`	the `mplus_submission_df` object to summarize
`refresh`	if `TRUE`, check the status of jobs for this object before printing
`...`	additional arguments, not currently used

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("none", "screen", "popup", "html", "latex", "markdown"),
  filename = "",
  keepCols,
  dropCols,
  sortBy = NULL,
  caption = "",
  display = FALSE,
  ...,
  include.rownames = FALSE
)
```

Arguments

modelList	A list of models returned from the extractModelSummaries function.
type	A character vector indicating the type of output format to be generated. One of: "none", "screen", "popup", "html", "latex", or "markdown". Screen results in a simple summary table being sent to the R console.
filename	The name of the file to be created. Can be an absolute or relative path. If filename is a relative path or just the filename, then it is assumed that the file resides in the working directory getwd(). Example: "Mplus Summary.html". By default, no filename is given, which results in the output being sent to the console. Note that currently, filename only has an effect for "html" and "latex".
keepCols	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, keepCols is: c("Title", "LL", "Parameters", "AIC", "AICC", "BIC", "RMSEA_Estimate"). Example: c("Title", "LL", "AIC", "CFI")
dropCols	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, dropCols is NULL. Example: c("InputInstructions", "TLI")

sortBy	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 NULL, which does not sort the table.
caption	A character string, the caption to be given to the table. Currently only applies to types "html", "latex", and "markdown".
display	optional logical (defaults to FALSE). This parameter specifies whether to display the table upon creation (TRUE or FALSE).
include.rownames	optional logical whether to include rownames or not.
...	additional arguments passed on to specific formatting types.

Value

Invisibly returns the summary table, which can be used if the printing options avaiable 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")
closeAllConnections()

## 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 `mplus.model` or `mplus.bparameters`). Obtained by SAVEDATA:BPARAMETERS in Mplus and `getSavedata_Bparams` or `readModels` in `MplusAutomation`.
test	The `R` code defining the parameter test of three or more parameters. Example: `"(STAITOT.ON.CG > STAITOT.ON.UCG) & (BDIM.ON.CG > BDIM.ON.UCG)"`.

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.UCG) & (BDIM.ON.CG > BDIM.ON.UCG)"`.

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.UCG) & (STDYX_BDIM.ON.CG > STDYX_BDIM.ON.UCG)")

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

## End(Not run)
```

---

testBParamConstraint *Test inequality-constrained hypothesis for two parameters based on iterations of MCMC chains*

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 `mplus.model` or `mplus.bparameters`). Obtained by SAVEDATA:BPARAMETERS in Mplus and `getSavedata_Bparams` or `readModels` in `MplusAutomation`.

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 >=, >, <, or <=. Example: ">="
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 limitted 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)
```

`update.mplusObject` *Update an Mplus model object*

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, quiet = TRUE, ...)
```

Arguments

object	An object of class mplusObject
quiet	optional. If TRUE, show status messages in the console.
...	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;"))

## check that use variables can be updated & overridden
str(update(example1, usevariables = c("mpg", "hp", "cyl")))

# 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