Package ‘multilevelmediation’

January 10, 2025

Type Package

Title Utility Functions for Multilevel Mediation Analysis

Version 0.4.1

Date 2024-12-28

Maintainer Carl F. Falk <carl.falk@mcgill.ca>

BugReports https://github.com/falkcarl/multilevelmediation/issues

Description The ultimate goal is to support 2-2-1, 2-1-1, and 1-1-1 models for multilevel mediation, the option of a moderating variable for either the a, b, or both paths, and covariates. Currently the 1-1-1 model is supported and several options of random effects; the initial code for bootstrapping was evaluated in simulations by Falk, Vogel, Hammami, and Miočević (2024) <doi:10.3758/s13428-023-02079-4>.
Support for Bayesian estimation using 'brms' comprises ongoing work. Currently only continuous mediators and outcomes are supported. Factors for any predictors must be numerically represented.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

Imports furrr, future, matrixcalc, MCMCpack, nlme, parallel, parallelly, stats, tidyr, brms, posterior, glmmTMB

Suggests testthat (>= 3.0.0), boot, BH

Depends R (>= 2.10)

Config/testthat/edition 3

NeedsCompilation no

Author Carl F. Falk [cre, aut],
Todd Vogel [aut],
Sarah Hammami [aut],
Milica Miočević [aut]

Repository CRAN

Date/Publication 2025-01-10 12:10:02 UTC

Contents

---

boot.modmed.mlm — *Boot function for (moderated) mediation with 2-level multilevel models*

Description

Boot function for (moderated) mediation with 2-level multilevel models

Usage

```
boot.modmed.mlm(
  data,
  indices,
  L2ID,
  ...,
  type = "all",
  modval1 = NULL,
  modval2 = NULL,
  boot.lvl = c("both", "1", "2")
)
```

Arguments

`data`	Data frame in long format.
`indices`	`boot` requires the function signature to accept a vector of index numbers and so this argument is required. If the index numbers are all in order starting at 1, then the relevant model will be fit to the data without any resampling. If some other vector is supplied, then resampling is done as described in details.
`L2ID`	Name of column that contains grouping variable in 'data' (e.g., "SubjectID")
`...`	Arguments passed to `modmed.mlm` to define the mediation analysis model.

`type`	Character that defines what information to extract from the model. Default and options are in `extract.modmed.mlm`. As examples, "indirect" will compute the indirect effect, "all" will save all random and fixed effects for possible additional computations, "indirect.diff" will compute the difference in the indirect effect at two values of a possible moderating variable.
`modval1`	(Optional) Numeric. If the model has a moderator, this value will be passed to `extract.modmed.mlm` to compute the indirect effect or other effects at that value. See `extract.modmed.mlm` for details.
`modval2`	(Optional). If the model has a moderator, it is possible to compute the difference in the indirect at two values of the moderator. If given and an appropriate option for such a difference is chosen for `type`, this value and that of `modval1` will be passed to `extract.modmed.mlm` to compute and save the difference. This is useful for obtaining a CI for the difference in the indirect effect at two different levels of the moderator.
`boot.lvl`	Character that defines at what level resampling should occur. Options are "both", "1", or "2". "both" will sample L2 units and then L1 units w/in each cluster. This has been noted to result in unequal sample sizes if the original clusters did not have equal sample sizes. "2" resamples only L2 units and leaves all L1 units intact. "1" will assume that whatever indices are fed from the boot function will be used. This probably only makes sense if `strata` is specified.

Details

Implements function to do bootstrapping with the 1-1-1 multilevel mediation analysis models as used in Falk, Vogel, Hammami & Miočević (in press). For use with boot package. This function aides in implementing case resampling methods with support for resampling at level 2, level 1, or both (e.g., see Hox and van de Schoot, 2013; van der Leeden, Meijer, & Busing, 2008). These functions also support moderated mediation. See also `modmed.mlm`. Note that `nlm` was used as the optimizer for some of the examples below as it was found to be faster for the models/simulations studied by Falk et al (in press).

Value

A vector of parameter estimates, depending on the value of `type` specified as input. Once the model is estimated, `extract.modmed.mlm is used to obtain the parameter estimates.`

References

Bauer, D. J., Preacher, K. J., & Gil, K. M. (2006). Conceptualizing and testing random indirect effects and moderated mediation in multilevel models: New procedures and recommendations. Psychological Methods, 11(2), 142–163. doi:10.1037/1082989X.11.2.142

Falk, C. F., Vogel, T., Hammami, S., & Miočević, M. (in press). Multilevel mediation analysis in R: A comparison of bootstrap and Bayesian approaches. Behavior Research Methods. doi:10.3758/s13428023020794 Preprint: doi:10.31234/osf.io/ync34

Hox, J., & van de Schoot, R. (2013). Robust methods for multilevel analysis. In M. A. Scott, J. S. Simonoff & B. D. Marx (Eds.), The SAGE Handbook of Multilevel Modeling (pp. 387-402). SAGE Publications Ltd. doi:10.4135/9781446247600.n22 van der Leeden, R., Meijer, E., & Busing, F. M. T. A. (2008). Resampling multilevel models. In J. de Leeuw & E. Meijer (Eds.), Handbook of Multilevel Analysis (pp. 401-433). Springer.

Examples

```
# Note that for all examples below, R should be increased to something
#  MUCH larger (e.g., 1000). Small values here are used only so that the code
#  runs relatively quickly when tested.

## Mediation for 1-1-1 model
data(BPG06dat)

#Setup parallel processing
# snow appears to work on Windows; something else may be better on Unix/Mac/Linux
library(parallel)
library(boot)
ncpu<-2
RNGkind("L'Ecuyer-CMRG") # set type of random number generation that works in parallel
cl<-makeCluster(ncpu)
clusterSetRNGStream(cl, 9912) # set random number seeds for cluster

# bootstrap all fixed and random effects (type="all")
boot.result<-boot(BPG06dat, statistic=boot.modmed.mlm, R=10,
  L2ID = "id", X = "x", Y = "y", M = "m",
  random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
  type="all",
  control=list(opt="nlm"),
  parallel="snow",ncpus=ncpu,cl=cl)

# Point estimate and 95% CI for indirect effect
extract.boot.modmed.mlm(boot.result, type="indirect", ci.conf=.95)

stopCluster(cl) # shut down cluster

## Moderated mediation

data(simdat)

# Note: use of nlm apparently fails in this moderated mediation model
# default optimizer for lme instead is used

## Bootstrap w/ moderation of a and b paths
set.seed(1234)

boot.result2<-boot(simdat, statistic=boot.modmed.mlm, R=5,
 L2ID = "L2id", X = "X", Y = "Y", M = "M",
   random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
   moderator = "mod", mod.a=TRUE, mod.b=TRUE,
   type="all")

# indirect effect point estimate and 95% CI when moderator = 0
```

```
extract.boot.modmed.mlm(boot.result2, type="indirect")
extract.boot.modmed.mlm(boot.result2, type="indirect", modval1=0)

# indirect effect point estimate and 95% CI when moderator = 1
extract.boot.modmed.mlm(boot.result2, type="indirect", modval1=1)

# indirect effect difference point estimate and 95% CI
extract.boot.modmed.mlm(boot.result2, type="indirect.diff",
   modval1=0, modval2=1)

# Example to not fail when using missing values
# See documentation for lme function from the nlme package for other
# options for na.action
dat.miss <- BPG06dat
dat.miss$m[c(1,2,3,4)]<-NA
dat.miss$y[c(5,6,7,8)]<-NA
boot.result<-boot(dat.miss, statistic=boot.modmed.mlm, R=5,
  L2ID = "id", X = "x", Y = "y", M = "m",
  random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
  type="all",
  control=list(opt="nlm"),
  na.action = na.omit)
```

boot.modmed.mlm.custom

Bootstrapping multilevel mediation model (without boot package)

Description

Bootstrapping multilevel mediation model (without boot package)

Usage

```
boot.modmed.mlm.custom(
  data,
  L2ID,
  ...,
  return.type = "all",
  modval1 = NULL,
  modval2 = NULL,
  nrep = 500,
  boot.type = c("caseboth", "case2", "case1", "resid"),
  parallel.type = c("lapply", "parallel", "furrr"),
  ncores = NULL,
  seed = NULL
)
```

Arguments

`data`	Data frame in long format. The function will do restructuring using `stack_bpg`.
`L2ID`	Name of column that contains grouping variable in 'data' (e.g., "SubjectID")
`...`	Arguments passed to `modmed.mlm` or `lme` to define the mediation analysis model or do estimation, respectively.
`return.type`	Character that defines what information to extract from the model. Default and options are in `extract.modmed.mlm`. As examples, "indirect" will compute the indirect effect, "all" will save all random and fixed effects for possible additional computations, "indirect.diff" will compute the difference in the indirect effect at two values of a possible moderating variable.
`modval1`	(Optional) Numeric. If the model has a moderator, this value will be passed to `extract.modmed.mlm` to compute the indirect effect or other effects at that value. See `extract.modmed.mlm` for details.
`modval2`	(Optional). If the model has a moderator, it is possible to compute the difference in the indirect at two values of the moderator. If given and an appropriate option for such a difference is chosen for `type`, this value and that of `modval1` will be passed to `extract.modmed.mlm` to compute and save the difference. This is useful for obtaining a CI for the difference in the indirect effect at two different levels of the moderator.
`nrep`	Number of bootstrap replications to perform. Pick a small number just for testing purposes, something larger (e.g., 1000 or more) for analyses.
`boot.type`	Character indicating the type of bootstrapping to perform. Options are: "caseboth", "case2", "case1", or "resid".
`parallel.type`	Character indicating type of parallel processing (if any) to use. Options are "lapply" (no parallel processing),"parallel" (uses `parallel` package), or "furrr" (uses `furrr` package.
`ncores`	Integer indicating the number of processing cores to attempt to use.
`seed`	Integer to set random number seed, for replication purposes. Note that replication may be somewhat R version or platform dependent.

Details

This function was written to do all four kinds of bootstrapping outlined in Falk, Vogel, Hammami & Miočević (in press): case resampling at both levels, at level 2 only, at level 1 only, and the residual-based bootstrap (e.g., see Hox and van de Schoot, 2013; van der Leeden, Meijer, & Busing, 2008). These functions also support moderated mediation. See also `modmed.mlm`. Note that `nlm` was used as the optimizer for some of the examples below as it was found to be faster for the models/simulations studied by Falk et al (in press). Note that Level 1 only bootstrapping is typically not recommended. See Falk et al. (in press) for details.

This function is different from the original functions used for the publication and that as of this writing still appear here: `boot.modmed.mlm` and here: `bootresid.modmed.mlm` . The present function seeks to unify case bootstrapping and residual-based bootstrapping in the same function. Furthermore, this newer function is also aimed at attempting to bypass the need for using the `boot` package to do computations and parallel processing. Some performance gains in terms of speed have been observed via use of this function instead of `boot` in conjunction with `boot.modmed.mlm`. Although somewhat slower, `furrr` can also be used if one would like a progress bar.

Value

A list with the following elements. Note that `t0` and `t` are intended to trick the `boot` package into working with some if its functions.

- `call` Call/arguments used when invoking this function. Useful for later extracting things like indirect effect.
- `t0` Parameter estimates based on the dataset.
- `t` Bootstrap distribution of all parameter estimates.
- `model` Fitted model to restructured data as one would obtain from `modmed.mlm`.
- `conv` Whether model fit to restructured dataset converged.
- `args` Arguments used when calling `modmed.mlm`. Useful for later extracting things like indirect effect.

References

Bauer, D. J., Preacher, K. J., & Gil, K. M. (2006). Conceptualizing and testing random indirect effects and moderated mediation in multilevel models: New procedures and recommendations. Psychological Methods, 11(2), 142–163. doi:10.1037/1082989X.11.2.142

Falk, C. F., Vogel, T., Hammami, S., & Miočević, M. (in press). Multilevel mediation analysis in R: A comparison of bootstrap and Bayesian approaches. Behavior Research Methods. doi:10.3758/s13428023020794 Preprint: doi:10.31234/osf.io/ync34

Hox, J., & van de Schoot, R. (2013). Robust methods for multilevel analysis. In M. A. Scott, J. S. Simonoff & B. D. Marx (Eds.), The SAGE Handbook of Multilevel Modeling (pp. 387-402). SAGE Publications Ltd. doi:10.4135/9781446247600.n22

van der Leeden, R., Meijer, E., & Busing, F. M. T. A. (2008). Resampling multilevel models. In J. de Leeuw & E. Meijer (Eds.), Handbook of Multilevel Analysis (pp. 401-433). Springer.

Examples

```
data(BPG06dat)

# Note that for all examples below, nrep should be increased to something
#  MUCH larger (e.g., 1000). Small values here are used only so that the code
#  runs relatively quickly when tested.

# double bootstrap, no parallel processing
boot.result<-boot.modmed.mlm.custom(BPG06dat, nrep=10, L2ID="id", X="x", Y="y", M="m",
  boot.type="caseboth",
  control=list(opt="nlm"), seed=1234)

extract.boot.modmed.mlm(boot.result, type="indirect", ci.conf=.95)

# residual bootstrap, parallel package
boot.result<-boot.modmed.mlm.custom(BPG06dat, nrep=10, L2ID="id", X="x", Y="y", M="m",
  boot.type="resid", random.a=TRUE, random.b=TRUE,
  parallel.type="parallel",ncores=2,seed=2299,
  control=list(opt="nlm"))

extract.boot.modmed.mlm(boot.result, type="indirect", ci.conf=.95)



# Example with moderation
data(simdat)

# moderation
boot.result<-boot.modmed.mlm.custom(simdat, nrep=5, L2ID = "L2id", X = "X", Y = "Y", M = "M",
   boot.type="caseboth",
   random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
   moderator = "mod", mod.a=TRUE, mod.b=TRUE,
   random.mod.a = TRUE, random.mod.b = TRUE,
  parallel.type="parallel",ncores=2,seed=2299)

extract.boot.modmed.mlm(boot.result, type="indirect")

# indirect effect point estimate and 95% CI when moderator = 0
extract.boot.modmed.mlm(boot.result, type="indirect", modval1=0)

# indirect effect point estimate and 95% CI when moderator = 1
 extract.boot.modmed.mlm(boot.result, type="indirect", modval1=1)

# indirect effect difference point estimate and 95% CI
extract.boot.modmed.mlm(boot.result, type="indirect.diff",
  modval1=0, modval2=1)
```

bootresid.modmed.mlm *Custom function for residual bootstrap for (moderated) multilevel mediation*

Description

Custom function for residual bootstrap for (moderated) multilevel mediation

Usage

```
bootresid.modmed.mlm(
  data,
  L2ID,
  R = 1000,
  X,
  Y,
  M,
  moderator = NULL,
  covars.m = NULL,
  covars.y = NULL,
  ...,
  type = "all",
  modval1 = NULL,
  modval2 = NULL
)
```

Arguments

`data`	Data frame in long format.
`L2ID`	Name of column that contains grouping variable in 'data' (e.g., "SubjectID")
`R`	Number of resamples
`X`	(Character) Name of column that contains the X independent variable in `data`.
`Y`	(Character) Name of column that contains the Y dependent variable in `data`.
`M`	(Character) Name of column that contains the M mediating variable in `data`.
`moderator`	Optional Character that contains name of column that contains the moderator variable in `data`
`covars.m`	(Character vector) Optional covariates to include in the model for M.
`covars.y`	(Character vector) Optional covariates to include in the model for Y.
`...`	Arguments passed to `modmed.mlm` to define the mediation analysis model.
`type`	Character that defines what information to extract from the model. Default and options are in `extract.modmed.mlm`. As examples, "indirect" will compute the indirect effect, "all" will save all random and fixed effects for possible additional computations, "indirect.diff" will compute the difference in the indirect effect at two values of a possible moderating variable.
`modval1`	(Optional) Numeric. If the model has a moderator, this value will be passed to `extract.modmed.mlm` to compute the indirect effect or other effects at that value. See `extract.modmed.mlm` for details.
`modval2`	(Optional). If the model has a moderator, it is possible to compute the difference in the indirect at two values of the moderator. If given and an appropriate option for such a difference is chosen for `type`, this value and that of `modval1` will be passed to `extract.modmed.mlm` to compute and save the difference. This is useful for obtaining a CI for the difference in the indirect effect at two different levels of the moderator.

Details

This function restructures data following Bauer, Pearcher, & Gil (2006) and then conducts residual-based bootstrapping in order to later obtain confidence intervals for the indirect effect and other coefficients. The residual-based bootstrap is described in Falk, Vogel, Hammami, & Miočević's manuscript (in press), but generally follows the procedure by Carpenter, Goldstein, & Rashbash (2003; See also Lai, 2021). Currently this function does not support parallel processing. See the newer `boot.modmed.mlm.custom` version for a re-write that does.

Value

A list with the following elements. Note that `t0` and `t` are intended to trick the `boot` package into working with some if its functions.

- `t0` Parameter estimates based on the dataset.
- `t` Bootstrap distribution of all parameter estimates.
- `model` Fitted model to restructured data as one would obtain from `modmed.mlm`.
- `call` Call/arguments used when invoking this function. Useful for later extracting things like indirect effect.

References

Bauer, D. J., Preacher, K. J., & Gil, K. M. (2006). Conceptualizing and testing random indirect effects and moderated mediation in multilevel models: new procedures and recommendations. Psychological Methods, 11(2), 142-163. doi:10.1037/1082989X.11.2.142

Carpenter, J. R., Goldstein, H., & Rasbash, J. (2003). A novel bootstrap procedure for assessing the relationship between class size and achievement. Applied Statistics, 52(4), 431-443.

Falk, C. F., Vogel, T., Hammami, S., & Miočević, M. (in press). Multilevel mediation analysis in R: A comparison of bootstrap and Bayesian approaches. Behavior Research Methods. doi:10.3758/s13428023020794 Preprint: doi:10.31234/osf.io/ync34

Lai, M. (2021). Bootstrap confidence intervals for multilevel standardized effect size. Multivariate Behavioral Research, 56(4), 558-578. doi:10.1080/00273171.2020.1746902

Examples

```
# Example data for 1-1-1 w/o moderation
data(BPG06dat)

# Note that R should be set to something MUCH larger, such as 1000 or greater.
# A low number here is chosen only so testing this example code goes relatively
# quickly
bootresid <- bootresid.modmed.mlm(BPG06dat,L2ID="id", X="x", Y="y", M="m",
  R=5, random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
  control=list(opt="nlm")
  )

extract.boot.modmed.mlm(bootresid, type="indirect")
```

`BPG06dat` *Simulated dataset from Bauer, Preacher, and Gil (2006)*

Description

In supplementary materials, Bauer, Preacher, and Gil (2006) provide a simulated dataset in SAS format. That dataset is replicated here.

Usage

```
BPG06dat
```

Format

A data frame with 800 observations (from 100 subjects) and 4 variables

id Level 2 or subject ID

x Predictor

m Mediator

y Outcome

Source

doi:10.1037/1082989X.11.2.142

References

Bauer, D. J., Preacher, K. J., & Gil, K. M. (2006). Conceptualizing and testing random indirect effects and moderated mediation in multilevel models: New procedures and recommendations. Psychological Methods, 11(2), 142–163. doi:10.1037/1082989X.11.2.142

`extract.boot.modmed.mlm`
Post-processing of bootstrap results from boot.modmed.mlm

Description

Post-processing of bootstrap results from boot.modmed.mlm

Usage

```
extract.boot.modmed.mlm(
  boot.obj,
  type = c("indirect", "a", "b", "cprime", "covab", "indirect.diff", "a.diff", "b.diff",
    "cprime.diff"),
  ci.type = "perc",
  ci.conf = 0.95,
  modval1 = NULL,
  modval2 = NULL
)
```

Arguments

`boot.obj`	Result of `boot` using `boot.modmed.mlm`
`type`	Character indicating which piece of information to extract from the model "indirect": value of the indirect effect. "a": Current value of a path. "b": Current value of b path. "cprime": Current value of c path. "covab": Random effect covariance between a and b paths, if both paths have associated random effects. "indirect.diff": difference in indirect effect at two values of the moderator (set by `modval1` and `modval2`). "a.diff": difference in a at two values of the moderator (set by `modval1` and `modval2`). "b.diff": difference in b at two values of the moderator (set by `modval1` and `modval2`). "cprime.diff": difference cprime at two values of the moderator (set by `modval1` and `modval2`).
`ci.type`	Character indicating the type of confidence interval to compute. Currently only percentile confidence intervals are supported with "perc".
`ci.conf`	Numeric value indicating the confidence level for the interval.
`modval1`	If enabled, other quantities such as the indirect effect, a, b, and cprime, will be computed at this particular value of the moderator. Otherwise, value of these quantities is directly extracted from the model output (i.e., these would represent values of the effects when the moderator = 0).
`modval2`	Second value of the moderator at which to compute the indirect effect.

Details

This is a convenience function that computes point estimates and confidence intervals from multilevel mediation analysis models where `boot.modmed.mlm` was used along with the `boot` package, or `bootresid.modmed.mlm` was used. This function generally assumes that type="all" was used when initially fitting the model, making all necessary information available for computation of indirect effects, differences between effects, and so on. If type="all" was not used, there is no guarantee that confidence intervals for the effects of interest can be extracted.

Value

A list with the following elements:

- `CI` A vector, typically two elements, that has the lower and upper endpoints requested confidence interval for the quantity requested by `type`.
- `est` Point estimate for the quantity requested by `type`.

Examples

```
## Mediation for 1-1-1 model
library(boot)

data(BPG06dat)

set.seed(1234)

# Note that R should be be MUCH larger than the value used here (e.g., 1000 or
# larger). A small number is chosen just so examples run relatively fast when
# tested.

# bootstrap all fixed and random effects
boot.result<-boot(BPG06dat, statistic=boot.modmed.mlm, R=5,
   L2ID = "id", X = "x", Y = "y", M = "m",
   random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
   type="all",
   control=list(opt="nlm"))

# Point estimate and 95% CI for indirect effect
extract.boot.modmed.mlm(boot.result, type="indirect", ci.conf=.95)
```

`extract.modmed.mlm` *Post-processing of a model fit with modmed.mlm*

Description

Post-processing of a model fit with modmed.mlm

Usage

```
extract.modmed.mlm(
  fit,
 type = c("all", "fixef", "recov", "recov.vec", "indirect", "a", "b", "cprime", "covab",
    "indirect.diff", "a.diff", "b.diff", "cprime.diff"),
  modval1 = NULL,
  modval2 = NULL
)
```

Arguments

`fit`	Result of `modmed.mlm`.
`type`	Character indicating which piece of information to extract from the model "all": fixed effects and var-cov matrix of random effects, as a single vector. "fixef": just fixed effects. "recov": var-cov matrix of random effects, as a matrix. "recov.vec": var-cov matrix of random effects, as a stacked vector. "indirect": value

	of the indirect effect. "a": Current value of a path. "b": Current value of b path. "cprime": Current value of c path. "covab": Random effect covariance between a and b paths, if both paths have associated random effects. "indirect.diff": difference in indirect effect at two values of the moderator (set by `modval1` and `modval2`). "a.diff": difference in a at two values of the moderator (set by `modval1` and `modval2`). "b.diff": difference in b at two values of the moderator (set by `modval1` and `modval2`). "cprime.diff": difference cprime at two values of the moderator (set by `modval1` and `modval2`).
`modval1`	If enabled, other quantities such as the indirect effect, a, b, and cprime, will be computed at this particular value of the moderator. Otherwise, value of these quantities is directly extracted from the model output (i.e., these would represent values of the effects when the moderator = 0).
`modval2`	Second value of the moderator at which to compute the indirect effect.

Details

This function extracts relevant parameter estimates from models estimated using `modmed.mlm`. For any of the .diff values, these are always the value of the effect at modval1 minus modval2.

Value

A vector or single numeric value corresponding to the parameter estimate(s) of interest is returned.

Examples

```
# Example data for 1-1-1 w/o moderation
data(BPG06dat)

# Fit model
fit<-modmed.mlm(BPG06dat,"id", "x", "y", "m",
  random.a=TRUE, random.b=TRUE, random.cprime=TRUE)
extract.modmed.mlm(fit, type="indirect")
```

`extract.modmed.mlm.brms`

Post-processing of results from modmed.mlm.brms

Description

Post-processing of results from modmed.mlm.brms

Usage

```
extract.modmed.mlm.brms(
  brms.obj,
  type = c("indirect", "a", "b", "cprime", "covab", "indirect.diff", "a.diff", "b.diff",
    "cprime.diff"),
  ci.type = c("ECI"),
  ci.conf = 0.95,
  modval1 = NULL,
  modval2 = NULL
)
```

Arguments

`brms.obj`	Result of `boot.modmed.mlm`
`type`	Character indicating which piece of information to extract from the model "indirect": value of the indirect effect. "a": Current value of a path. "b": Current value of b path. "cprime": Current value of c path. "covab": Random effect covariance between a and b paths, if both paths have associated random effects. "indirect.diff": difference in indirect effect at two values of the moderator (set by `modval1` and `modval2`). "a.diff": difference in a at two values of the moderator (set by `modval1` and `modval2`). "b.diff": difference in b at two values of the moderator (set by `modval1` and `modval2`). "cprime.diff": difference cprime at two values of the moderator (set by `modval1` and `modval2`).
`ci.type`	Character indicating the type of confidence interval to compute. For now, just "ECI" is supported, an equal-tailed credible interval.
`ci.conf`	Numeric value indicating the confidence level for the credibility interval.
`modval1`	If enabled, other quantities such as the indirect effect, a, b, and cprime, will be computed at this particular value of the moderator. Otherwise, value of these quantities is directly extracted from the model output (i.e., these would represent values of the effects when the moderator = 0).
`modval2`	Second value of the moderator at which to compute the indirect effect.

Details

This function generally assumes that type="all" was used when initially fitting the model, making all necessary information available for computation of indirect effects, differences between effects, and so on. If type="all" was not used, there is no guarantee that credibility intervals for the effects of interest can be extracted.

Value

A list with two elements:

- `CI` Point estimate (mean and median of posterior), sd, mad, credibility interval (quantiles), and other diagnostic information (rhat, ess_bulk, ess_tail).
- `draws` Contains `draws_matrix` (from the posterior package) for quantity of interest. i.e., all posterior draws, for which the user may do additional work with.

Examples

```
data(BPG06dat)

# Note: 2000 iterations is just an example so that run time is not too long.
# Pick something larger (e.g., 5000+) in practice

# Only fixed effects with random intercept
fit<-modmed.mlm.brms(BPG06dat,"id", "x", "y" , "m", cores = 2,
                     iter = 2000, control = list(adapt_delta=0.95),
                     seed = 1234)


res.indirect <- extract.modmed.mlm.brms(fit, "indirect")
res.a <- extract.modmed.mlm.brms(fit, "a")
res.b <- extract.modmed.mlm.brms(fit, "b")
res.cprime <- extract.modmed.mlm.brms(fit, "cprime")

# Summary of results is in CI slot, example.
# Here, 95% credibility interval is denoted by q2.5 and q97.5
res.indirect$CI

# Matrix of draws in another slot:
res.indirect$draws
```

`modmed.mlm` *Model definition and estimation function for two-level (moderated) mediation*

Description

Model definition and estimation function for two-level (moderated) mediation

Usage

```
modmed.mlm(
  data,
  L2ID,
  X,
  Y,
  M,
  moderator = NULL,
  mod.a = FALSE,
  mod.b = FALSE,
  mod.cprime = FALSE,
  covars.m = NULL,
  covars.y = NULL,
  random.a = FALSE,
  random.b = FALSE,
  random.cprime = FALSE,
  random.mod.a = FALSE,
  random.mod.b = FALSE,
  random.mod.cprime = FALSE,
  random.mod.m = FALSE,
  random.mod.y = FALSE,
  random.covars.m = NULL,
  random.covars.y = NULL,
  random.int.m = TRUE,
  random.int.y = TRUE,
  estimator = c("lme", "glmmTMB"),
  method = c("REML", "ML"),
  control = NULL,
  returndata = FALSE,
  datmfun = NULL,
  data.stacked = NULL,
  ...
)
```

Arguments

`data`	Data frame in long format.
`L2ID`	(Character) Name of column that contains grouping variable in `data` (e.g., `"SubjectID"`).
`X`	(Character) Name of column that contains the X independent variable in `data`.
`Y`	(Character) Name of column that contains the Y dependent variable in `data`.
`M`	(Character) Name of column that contains the M mediating variable in `data`.
`moderator`	Optional Character that contains name of column that contains the moderator variable in `data`
`mod.a`	(Logical) Add moderator to 'a' path (i.e., SmX:W, where W is the moderator)?
`mod.b`	(Logical) Add moderator to 'b' path (i.e., SyM:W, where W is the moderator)?
`mod.cprime`	(Logical) Add moderator to 'c' path (i.e., SyX:W, where W is the moderator)
`covars.m`	(Character vector) Optional covariates to include in the model for M.
`covars.y`	(Character vector) Optional covariates to include in the model for Y.
`random.a`	(Logical) Add random slope for 'a' path (i.e,. SmX)?
`random.b`	(Logical) Add random slope for 'b' path (i.e., SyM)?
`random.cprime`	(Logical) Add random slope for 'cprime' direct effect path (i.e., SyX)?
`random.mod.a`	(Logical) Add random slope for 'a' path moderator?
`random.mod.b`	(Logical) Add random slope for 'b' path moderator?
`random.mod.cprime`	(Logical) Add random slope for 'c' path moderator?
`random.mod.m`	(Logical) Add random slope for effect of moderator on M?

`random.mod.y`	(Logical) Add random slope for effect of moderator on Y?
`random.covars.m`	(Character vector) If any covariates specified from `covars.m` are present, random effects are added for them.
`random.covars.y`	(Character vector) If any covariates specified from `covars.y` are present, random effects are added for them.
`random.int.m`	(Logical) Add random intercept for M? (defaults to TRUE)
`random.int.y`	(Logical) Add random intercept for Y? (defaults to TRUE)
`estimator`	(Character) Which program to use to estimate models? `lme` is what was originally tested with the package and publication, but support for `glmmTMB` is now available.
`method`	Argument used to control estimation method. Options are "REML" (default) or "ML".
`control`	Argument passed to `lme` or `glmmTMB` that controls other estimation options. See those functions for the `control` argument. If `lme` is chosen for estimation, but nothing is specified for `control`, some defaults values are populated that basically greatly increase the number of admissible iterations.
`returndata`	(Logical) Whether to save restructured data in its own slot. Note: nlme may do this automatically. Defaults to `FALSE`.
`datmfun`	(experimental) A function that will do additional data manipulation on the restacked dataset. The function ought to take the restacked dataset (e.g., done using `stack_bpg`) and return a dataset that can be analyzed using `modmed.mlm` Could be used for some kind of additional centering strategy after data are restacked (and after bootstrapped) or some other missing data handling strategy. Either suggestion requires further study.
`data.stacked`	(experimental) Currently used internally by bootresid.modmed.mlm to feed already stacked data to the function.
`...`	Pass any additional options down to `link[nlme]{lme}`. Added to handle missing values. e.g., `na.action = na.omit`.

Details

Implements custom function to do 1-1-1 multilevel mediation model following Bauer, Preacher, & Gil (2006). The basic procedure involves restructuring the data (`stack_bpg`) and then estimating the model using `lme`. The model assumes heteroscedasticity since the mediator and outcome variable may have different error variances. The function also supports covariates as predictors of the mediator and/or outcome, as well as moderated mediation. Currently a single moderator variable is supported and it may moderate any/all paths of the model. However, the the moderator is assumed continuous. While it may be possible to include moderators that are categorical, it is not currently automated (i.e., the user will need to manually code the categorical variable as numeric).

For more information for variable labels and how these will correspond to the output coefficients, see the documentation for `stack_bpg`, as those docs contain a description of all of the variables.

Value

A list with the following elements:

- `model` The fitted model using `lme`. Use as you would a fitted model from that package.
- `args` Arguments used to call the function. Useful for later automating extraction of the indirect effect or other quantities.
- `conv` Whether estimation appeared to converge.
- `data` If you asked for the restructured dataset to be returned, it shall be here.

References

Bauer, D. J., Preacher, K. J., & Gil, K. M. (2006). Conceptualizing and testing random indirect effects and moderated mediation in multilevel models: New procedures and recommendations. Psychological Methods, 11(2), 142–163. doi:10.1037/1082989X.11.2.142

Examples

```
# Example data for 1-1-1 w/o moderation
data(BPG06dat)

# Fit model
fit<-modmed.mlm(BPG06dat,"id", "x", "y", "m",
  random.a=TRUE, random.b=TRUE, random.cprime=TRUE)

extract.modmed.mlm(fit)
extract.modmed.mlm(fit, type="indirect")
extract.modmed.mlm(fit, type="a")
extract.modmed.mlm(fit, type="b")
extract.modmed.mlm(fit, type="covab")

# Vector of parameter estimates, including indirect effect
#fit$pars

# The saved, fitted model following Bauer, Preacher, & Gil (2006)
summary(fit$model)



# Fit model with moderation
data(simdat)

# moderation for a path
fitmoda<-modmed.mlm(simdat,"L2id", "X", "Y", "M",
  random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
  moderator = "mod", mod.a=TRUE)

# moderation for b path
fitmodb<-modmed.mlm(simdat,"L2id", "X", "Y", "M",
  random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
  moderator = "mod", mod.b=TRUE)
```

```
# moderation for both a and b paths
fitmodab<-modmed.mlm(simdat,"L2id", "X", "Y", "M",
  random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
  moderator = "mod", mod.a=TRUE, mod.b=TRUE)

# moderation for both a and b paths and random effect for interaction a
fitmodab2<-modmed.mlm(simdat,"L2id", "X", "Y", "M",
  random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
  moderator = "mod", mod.a=TRUE, mod.b=TRUE,
  random.mod.a = TRUE, random.mod.m = TRUE)

# moderation for both a and b paths and random effect for interaction b
fitmodab3<-modmed.mlm(simdat,"L2id", "X", "Y", "M",
  random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
  moderator = "mod", mod.a=TRUE, mod.b=TRUE,
  random.mod.b = TRUE, random.mod.y = TRUE)

# moderation for both a and b paths and random effect for both interactions
fitmodab4<-modmed.mlm(simdat,"L2id", "X", "Y", "M",
  random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
  moderator = "mod", mod.a=TRUE, mod.b=TRUE,
  random.mod.a = TRUE, random.mod.b = TRUE,
  random.mod.m = TRUE, random.mod.y = TRUE)

# compare models?
# Apparently anova() is not supported as it's looking for fixed.formula,
# as it's not in the current environment
# AIC works though
AIC(fitmodab$model)
AIC(fitmodab2$model)
AIC(fitmodab3$model)
AIC(fitmodab4$model) # AIC here is best. Great simulated data we have here

extract.modmed.mlm(fitmodab4, "indirect")
extract.modmed.mlm(fitmodab4, "indirect", modval1=0) # should match above
extract.modmed.mlm(fitmodab4, "indirect", modval1=1)
extract.modmed.mlm(fitmodab4, "indirect.diff", modval1 = 0, modval2=1)
extract.modmed.mlm(fitmodab4, "indirect", modval1=0)-
  extract.modmed.mlm(fitmodab4, "indirect", modval1=1) # should match prev line

extract.modmed.mlm(fitmodab4, "a")
extract.modmed.mlm(fitmodab4, "a", modval1=0) # should match above
extract.modmed.mlm(fitmodab4, "a", modval1=1)
extract.modmed.mlm(fitmodab4, "a.diff", modval1 = 0, modval2=1)
extract.modmed.mlm(fitmodab4, "a", modval1=0)-
  extract.modmed.mlm(fitmodab4, "a", modval1=1)  # should match prev line

extract.modmed.mlm(fitmodab4, "b")
extract.modmed.mlm(fitmodab4, "b", modval1=0) # should match above
extract.modmed.mlm(fitmodab4, "b", modval1=1)
extract.modmed.mlm(fitmodab4, "b.diff", modval1 = 0, modval2=1)
extract.modmed.mlm(fitmodab4, "b", modval1=0)-
```

```
    extract.modmed.mlm(fitmodab4, "b", modval1=1)  # should match prev line

  extract.modmed.mlm(fitmodab3, "indirect")
  extract.modmed.mlm(fitmodab3, "indirect", modval1=0) # should match above
  extract.modmed.mlm(fitmodab3, "indirect", modval1=1)
  extract.modmed.mlm(fitmodab3, "indirect.diff", modval1 = 0, modval2=1)
  extract.modmed.mlm(fitmodab3, "indirect", modval1=0)-
    extract.modmed.mlm(fitmodab3, "indirect", modval1=1) # should match prev line

  extract.modmed.mlm(fitmodab3, "a")
  extract.modmed.mlm(fitmodab3, "a", modval1=0) # should match above
  extract.modmed.mlm(fitmodab3, "a", modval1=1)
  extract.modmed.mlm(fitmodab3, "a.diff", modval1 = 0, modval2=1)
  extract.modmed.mlm(fitmodab3, "a", modval1=0)-
    extract.modmed.mlm(fitmodab3, "a", modval1=1)  # should match prev line

  extract.modmed.mlm(fitmodab3, "b")
  extract.modmed.mlm(fitmodab3, "b", modval1=0) # should match above
  extract.modmed.mlm(fitmodab3, "b", modval1=1)
  extract.modmed.mlm(fitmodab3, "b.diff", modval1 = 0, modval2=1)
  extract.modmed.mlm(fitmodab3, "b", modval1=0)-
    extract.modmed.mlm(fitmodab3, "b", modval1=1)  # should match prev line

  extract.modmed.mlm(fitmodab2, "indirect")
  extract.modmed.mlm(fitmodab2, "indirect", modval1=0) # should match above
  extract.modmed.mlm(fitmodab2, "indirect", modval1=1)
  extract.modmed.mlm(fitmodab2, "indirect.diff", modval1 = 0, modval2=1)
  extract.modmed.mlm(fitmodab2, "indirect", modval1=0)-
    extract.modmed.mlm(fitmodab2, "indirect", modval1=1) # should match prev line

  extract.modmed.mlm(fitmodab2, "a")
  extract.modmed.mlm(fitmodab2, "a", modval1=0) # should match above
  extract.modmed.mlm(fitmodab2, "a", modval1=1)
  extract.modmed.mlm(fitmodab2, "a.diff", modval1 = 0, modval2=1)
  extract.modmed.mlm(fitmodab2, "a", modval1=0)-
    extract.modmed.mlm(fitmodab2, "a", modval1=1)  # should match prev line

  extract.modmed.mlm(fitmodab2, "b")
  extract.modmed.mlm(fitmodab2, "b", modval1=0) # should match above
  extract.modmed.mlm(fitmodab2, "b", modval1=1)
  extract.modmed.mlm(fitmodab2, "b.diff", modval1 = 0, modval2=1)
  extract.modmed.mlm(fitmodab2, "b", modval1=0)-
    extract.modmed.mlm(fitmodab2, "b", modval1=1)  # should match prev line

  extract.modmed.mlm(fitmodab, "indirect")
  extract.modmed.mlm(fitmodab, "indirect", modval1=0) # should match above
  extract.modmed.mlm(fitmodab, "indirect", modval1=1)
  extract.modmed.mlm(fitmodab, "indirect.diff", modval1 = 0, modval2=1)
  extract.modmed.mlm(fitmodab, "indirect", modval1=0)-
    extract.modmed.mlm(fitmodab, "indirect", modval1=1) # should match prev line

  extract.modmed.mlm(fitmodab, "a")
  extract.modmed.mlm(fitmodab, "a", modval1=0) # should match above
```

```
extract.modmed.mlm(fitmodab, "a", modval1=1)
extract.modmed.mlm(fitmodab, "a.diff", modval1 = 0, modval2=1)
extract.modmed.mlm(fitmodab, "a", modval1=0)-
  extract.modmed.mlm(fitmodab, "a", modval1=1)  # should match prev line

extract.modmed.mlm(fitmodab, "b")
extract.modmed.mlm(fitmodab, "b", modval1=0) # should match above
extract.modmed.mlm(fitmodab, "b", modval1=1)
extract.modmed.mlm(fitmodab, "b.diff", modval1 = 0, modval2=1)
extract.modmed.mlm(fitmodab, "b", modval1=0)-
  extract.modmed.mlm(fitmodab, "b", modval1=1)  # should match prev line


# Example to not fail when using missing values
# Missing data handling is not that great as not all info is used, but is typical
# of default missing data handling strategies in MLM. See documentation for
# lme function in the nlme package for more options for na.action
dat.miss <- BPG06dat
dat.miss$m[c(1,2,3,4)]<-NA
dat.miss$y[c(5,6,7,8)]<-NA
fit<-modmed.mlm(dat.miss,"id", "x", "y", "m",
                random.a=TRUE, random.b=TRUE, random.cprime=TRUE,
                na.action = na.omit)
```

modmed.mlm.brms — *Custom model fitting function for a 1-1-1 (moderated) mediation for the brms code*

Description

Custom model fitting function for a 1-1-1 (moderated) mediation for the brms code

Usage

```
modmed.mlm.brms(
  data,
  L2ID,
  X,
  Y,
  M,
  moderator = NULL,
  mod.a = FALSE,
  mod.b = FALSE,
  mod.cprime = FALSE,
  covars.m = NULL,
  covars.y = NULL,
  random.a = FALSE,
  random.b = FALSE,
  random.cprime = FALSE,
  random.mod.a = FALSE,
  random.mod.b = FALSE,
  random.mod.cprime = FALSE,
  random.mod.m = FALSE,
  random.mod.y = FALSE,
  random.covars.m = NULL,
  random.covars.y = NULL,
  random.int.m = TRUE,
  random.int.y = TRUE,
  returndata = FALSE,
  family = gaussian,
  iter = 7000,
  control = list(adapt_delta = 0.95),
  chains = 4,
  ...
)
```

Arguments

`data`	Data frame in long format.
`L2ID`	(Character) Name of column that contains grouping variable in `data` (e.g., `"SubjectID"`).
`X`	(Character) Name of column that contains the X independent variable in `data`.
`Y`	(Character) Name of column that contains the Y dependent variable in `data`.
`M`	(Character) Name of column that contains the M mediating variable in `data`.
`moderator`	Optional Character that contains name of column that contains the moderator variable in `data`
`mod.a`	(Logical) Add moderator to 'a' path (i.e., SmX:W, where W is the moderator)?
`mod.b`	(Logical) Add moderator to 'b' path (i.e., SyM:W, where W is the moderator)?
`mod.cprime`	(Logical) Add moderator to 'c' path (i.e., SyX:W, where W is the moderator)
`covars.m`	(Character vector) Optional covariates to include in the model for M.
`covars.y`	(Character vector) Optional covariates to include in the model for Y.
`random.a`	(Logical) Add random slope for 'a' path (i.e,. SmX)?
`random.b`	(Logical) Add random slope for 'b' path (i.e., SyM)?
`random.cprime`	(Logical) Add random slope for 'cprime' direct effect path (i.e., SyX)?
`random.mod.a`	(Logical) Add random slope for 'a' path moderator?
`random.mod.b`	(Logical) Add random slope for 'b' path moderator?
`random.mod.cprime`	(Logical) Add random slope for 'c' path moderator?
`random.mod.m`	(Logical) Add random slope for effect of moderator on M?
`random.mod.y`	(Logical) Add random slope for effect of moderator on Y?
`random.covars.m`	(Logical vector) Add random slopes for covariates on M?

`random.covars.y`	(Logical vector) Add random slopes for covariates on Y?
`random.int.m`	(Logical) Add random intercept for M? (defaults to TRUE)
`random.int.y`	(Logical) Add random intercept for Y? (defaults to TRUE)
`returndata`	(Logical) Whether to save restructured data in its own slot. Defaults to `FALSE`.
`family`	Argument passed to `brm` A character string naming the distribution of the response variable to be used in the model.
`iter`	Argument passed to `brm` Number of total iterations.
`control`	Argument passed to `brm` To decrease (or eliminate at best) the number of divergent transitions that cause a bias in the obtained posterior samples.
`chains`	Argument passed to `brm` Set the number of chains
`...`	Additional arguments to pass to `brm`

Details

Implements custom function to do (moderated) mediation with two-level multilevel models with Bayesian estimation via the `brms` package. Does not handle covariates at the moment. Bayesian estimation using `brms` was studied by Falk, Vogel, Hammami & Miočević (in press). It is suggested if you use this function that you also do `cite("brms")` to figure out how to cite that package.

Value

A list with the following elements:

- `model` The fitted model from `brm`. Use as you would a fitted model from that package.
- `args` Arguments used to call the function. Useful for later automating extraction of the indirect effect or other quantities.
- `conv` Whether `brm` finished estimation, not diagnostic of convergence.

References

Falk, C. F., Vogel, T., Hammami, S., & Miočević, M. (in press). Multilevel mediation analysis in R: A comparison of bootstrap and Bayesian approaches. Behavior Research Methods. doi:10.3758/s13428023020794 Preprint: doi:10.31234/osf.io/ync34

Paul-Christian Bürkner (2017). brms: An R Package for Bayesian Multilevel Models Using Stan. Journal of Statistical Software, 80(1), 1-28. doi:10.18637/jss.v080.i01

Examples

```
# Note: 2000 iterations is just an example so that run time is not too long.
# Pick something larger (e.g., 5000+) in practice

# Example data for 1-1-1 w/o moderation
data(BPG06dat)

# random effects for a and b paths (and intercept), no moderation
# (For moderation, note that modmed.mlm syntax is typically the same)
fit<-modmed.mlm.brms(BPG06dat,"id", "x", "y" , "m", cores=2,
                     random.a=TRUE, random.b=TRUE,
                     iter = 2000, control = list(adapt_delta=0.95),
                     seed = 1234)

# Examine model results and some diagnostics
summary(fit$model)

# Potential scale reduction (PSR) or Rhat guidelines vary but the largest
#  should be close to 1 ( < 1.1, < 1.05, < 1.01).
# It is also possible to extract all of them.
max(brms::rhat(fit$model)) # largest rhat

# Fit (loo and WAIC)
brms::loo(fit$model)
brms::waic(fit$model)

# Point and interval estimates, diagnostics, for quantities of interest

# Traceplots: TODO, list conversions for how brms represents parameters with
# How these are colloquially referred to in mediation literature.
plot(fit$model, variable="b_SmX") # this is traceplot for one parameter

# Example of extracting/computing intervals for particular quantities
res.indirect <- extract.modmed.mlm.brms(fit, "indirect")
res.a <- extract.modmed.mlm.brms(fit, "a")
res.b <- extract.modmed.mlm.brms(fit, "b")
res.cprime <- extract.modmed.mlm.brms(fit, "cprime")

# Summary of results is in CI slot, example:
res.indirect$CI

# 99% CI
res.indirect <- extract.modmed.mlm.brms(fit, "indirect", ci.conf = .99)
```

simdat *A simulated dataset with moderator*

Description

This simulated dataset contains a hypothetical moderating variable that is dichotomously coded. The true model dictated that the moderator moderated both a and b paths of the mediational model, though the strength of moderation may be small.

Usage

```
simdat
```

Format

A data frame with 800 observations (from 50 subjects) and 5 variables

L2id Level 2 or subject ID

X Predictor

M Mediator

Y Outcome

mod Moderator

`stack_bpg`	*Stacks data in the style of Bauer-Preacher-Gil for multilevel mediation*

Description

Stacks data in the style of Bauer-Preacher-Gil for multilevel mediation

Usage

```
stack_bpg(data, L2ID, X, Y, M,
  moderator = NULL,
  covars.m = NULL,
  covars.y = NULL)
```

Arguments

`data`	Data frame in long format.
`L2ID`	(String) Name of column that contains grouping variable in `data` (e.g., `"SubjectID"`).
`X`	(String) Name of column that contains the X independent variable in `data`.
`Y`	(String) Name of column that contains the Y dependent variable in `data`.
`M`	(String) Name of column that contains the M mediating variable in `data`.
`moderator`	Optional Character that contains name of column that contains the moderator variable in `data`
`covars.m`	(Character vector) Optional covariates to include in the model for M.
`covars.y`	(Character vector) Optional covariates to include in the model for Y.

Details

This is a convenience function used primarily internally by the package to restructure data in the style of Bauer, Preacher, and Gil (2006). The point is to allow both Y and M to be outcomes in a single column ("Z"), so that both mediator and outcome models can be fit at the same time. This is necessary to estimate the covariance between "a" and "b" paths at the same time when both have random effects. Two selector variables, "Sy" and "Sm" toggle whether each row corresponds to the outcome or the mediator, respectively.

Value

An object that is a subclass of `data.frame` is returned. In particular a `tbl_df` or "tibble."

So that coefficients extracted later from `modmed.mlm` hopefully make more sense, the below lists all variables (i.e., typical column names) for the data frame.

- `X` Independent variable.
- `L2id` Level 2 ID variable.
- `Md` Value of the mediator (not necessarily used due to restructuring, however).
- `Outcome` Whether the row corresponds to M or Y as the outcome.
- `Z` Value of the outcome variable.
- `Sy` Indicator variable for Y as outcome. 0 if Y is not outcome for this row. 1 if Y is outcome for this row. In model output, this is the intercept for Y.
- `Sm` Indicator variable for M as outcome. 0 if M is not outcome for this row. 1 if M is outcome for this row. In model output, this is the intercept for M.
- `SmX` Value of X when M is the outcome (literally X times Sm); will be 0 when Y is outcome. In model output, this is the "a" path.
- `SyX` Value of X when Y is the outcome (literally X times Sy); will be 0 when M is outcome. In model output, this is the "cprime" path (direct effect).
- `SyM` Calue of M when Y is the outcome (literally M times Sy); will be 0 when M is the outcome. In model output, this is the "b" path.
- `W` Value of any moderating variable. This may show up in output later on as "SmX:W" (interaction with "a" path) or "SyM:W" (interaction with "b" path) or "SyX:W" (interaction with "cprime" path) depending on which path it moderates.
- If `covars.m` or `covars.y` are not null, any additional covariates will also be added to the data frame and their original names will be retained.

When `modmed.mlm` is used, any output with an "re" prefix will correspond to a random effect. Note that estimation with brms will result in slightly different output than listed here. Coefficients typically have a "b_" prefix, and random effects are parameterized not such that we end up with covariances, but using correlations and standard deviations for each effect.

References

Bauer, D. J., Preacher, K. J., & Gil, K. M. (2006). Conceptualizing and testing random indirect effects and moderated mediation in multilevel models: new procedures and recommendations. Psychological Methods, 11(2), 142-163. doi:10.1037/1082989X.11.2.142

Examples

```
# restructure BPG data
data(BPG06dat)

dat <- stack_bpg(BPG06dat,
  "id", "x", "y", "m"
)
```

```
head(dat)

# restructure simulated data w/ moderator
data(simdat)
dat2 <- stack_bpg(simdat,
  "L2id", "X", "Y", "M",
  moderator = "mod"
)

head(dat2)
```

Index