

Package ‘modsem’

July 7, 2024

Type Package

Title Latent Interaction (and Moderation) Analysis in Structural Equation Models (SEM)

Version 1.0.1

Maintainer Kjell Solem Slupphaug <slupphaugkjell@gmail.com>

Description Estimation of interaction (i.e., moderation) effects between latent variables in structural equation models (SEM).

The supported methods are:

The constrained approach (Algina & Moulder, 2001).

The unconstrained approach (Marsh et al., 2004).

The residual centering approach (Little et al., 2006).

The double centering approach (Lin et al., 2010).

The latent moderated structural equations (LMS) approach (Klein & Moosbrugger, 2000).

The quasi-

maximum likelihood (QML) approach (Klein & Muthén, 2007) (temporarily unavailable)

The constrained- unconstrained, residual- and double centering- approaches are estimated via 'lavaan' (Rosseel, 2012), whilst the LMS- and QML- approaches are estimated via by modsem it self. Alternatively model can be estimated via 'Mplus' (Muthén & Muthén, 1998-2017).

References:

Algina, J., & Moulder, B. C. (2001).

<doi:10.1207/S15328007SEM0801_3>.

`` A note on estimating the Jöreskog-

Yang model for latent variable interaction using 'LISREL' 8.3."

Klein, A., & Moosbrugger, H. (2000).

<doi:10.1007/BF02296338>.

`` Maximum likelihood estimation of latent interaction effects with the LMS method."

Klein, A. G., & Muthén, B. O. (2007).

<doi:10.1080/00273170701710205>.

`` Quasi-maximum likelihood estimation of structural equation models with multiple interaction and quadratic effects."

Lin, G. C., Wen, Z., Marsh, H. W., & Lin, H. S. (2010).

<doi:10.1080/10705511.2010.488999>.

`` Structural equation models of latent interactions: Clarification of orthogonalizing and double-mean-centering strategies."

Little, T. D., Bovaird, J. A., & Widaman, K. F. (2006).
[doi:10.1207/s15328007sem1304_1](https://doi.org/10.1207/s15328007sem1304_1).
 ``On the merits of orthogonalizing powered and product terms: Implications for modeling interactions among latent variables."
 Marsh, H. W., Wen, Z., & Hau, K. T. (2004).
[doi:10.1037/1082-989X.9.3.275](https://doi.org/10.1037/1082-989X.9.3.275).
 ``Structural equation models of latent interactions: evaluation of alternative estimation strategies and indicator construction."
 Muthén, L.K. and Muthén, B.O. (1998-2017).
 ``Mplus' User's Guide. Eighth Edition."
<https://www.statmodel.com/>.
 Rosseel Y (2012).
[doi:10.18637/jss.v048.i02](https://doi.org/10.18637/jss.v048.i02).
 ``lavaan': An R Package for Structural Equation Modeling."

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, purrr, stringr, lavaan, rlang, MplusAutomation, nlme,
 R6, dplyr, mvnfast, stats, fastGHQuad, mvtnorm, ggplot2

Depends R (>= 3.50)

URL <https://github.com/Kss2k/modsem>

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Kjell Solem Slupphaug [aut, cre]
 (<https://orcid.org/0009-0005-8324-2834>)

Repository CRAN

Date/Publication 2024-07-07 16:20:02 UTC

Contents

compare_fit	3
default_settings_da	4
default_settings_pi	5
extract_lavaan	5
fit_modsem_da	6
get_pi_data	6
get_pi_syntax	7
jordan	8
modsem	10
modsemify	12

modsem_da	13
modsem_inspect	16
modsem_mplus	17
modsem_pi	18
multiplyIndicatorsCpp	21
oneInt	22
parameter_estimates	22
plot_interaction	22
standardized_estimates	24
summary.modsem_lms	25
TPB	27
TPB_UK	28
trace_path	28
var_interactions	30

Index**31**

compare_fit	<i>compare model fit for qml and lms models</i>
-------------	---

Description

Compare the fit of two models using the likelihood ratio test. ‘estH0’ representing the null hypothesis model, and ‘estH1’ the alternative hypothesis model. Importantly, the function assumes that ‘estH0’ does not have more free parameters (i.e., degrees of freedom) than ‘estH1’. alternative hypothesis model

Usage

```
compare_fit(estH0, estH1)
```

Arguments

estH0	object of class ‘modsem_lms’ or ‘modsem_qml’ representing the null hypothesis model
estH1	object of class ‘modsem_lms’ or ‘modsem_qml’ representing the

Examples

```
## Not run:
H0 <- "
# Outer Model
X =~ x1 + x2 + x3
Y =~ y1 + y2 + y3
Z =~ z1 + z2 + z3

# Inner model
Y ~ X + Z
"
```

```
estH0 <- modsem(m1, oneInt, "lms")

H1 <- "
# Outer Model
X =~ x1 + x2 + x3
Y =~ y1 + y2 + y3
Z =~ z1 + z2 + z3

# Inner model
Y ~ X + Z + X:Z
"

estH1 <- modsem(m1, oneInt, "lms")
compare_fit(estH0, estH1)

## End(Not run)
```

default_settings_da *default arguments fro LMS and QML approach*

Description

This function returns the default settings for the LMS and QML approach.

Usage

```
default_settings_da(method = c("lms", "qml"))
```

Arguments

method which method to get the settings for

Value

modsem_lms or modsem_qml object

Examples

```
library(modsem)
default_settings_da()
```

default_settings_pi *default arguments for product indicator approaches*

Description

This function returns the default settings for the product indicator approaches

Usage

```
default_settings_pi(method = c("rca", "uca", "pind", "dblcent", "ca"))
```

Arguments

method which method to get the settings for

Value

modsem_lms or modsem_qml object

Examples

```
library(modsem)
default_settings_pi()
```

extract_lavaan *extract lavaan object from modsem object estimated using product indicators*

Description

extract lavaan object from modsem object estimated using product indicators

Usage

```
extract_lavaan(object)
```

Arguments

object modsem object

Value

lavaan object

Examples

```

library(modsem)
m1 <- '
  # Outer Model
  X =~ x1 + x2 + x3
  Y =~ y1 + y2 + y3
  Z =~ z1 + z2 + z3

  # Inner model
  Y ~ X + Z + X:Z
'
est <- modsem_pi(m1, oneInt)
lav_est <- extract_lavaan(est)

```

fit_modsem_da

Fit measures for QML and LMS models

Description

Calculates chi-sq test and p-value, as well as RMSEA for the LMS and QML models. Note that the Chi-Square based fit measures should be calculated for the baseline model, i.e., the model without the interaction effect

Usage

```
fit_modsem_da(model, chisq = TRUE)
```

Arguments

model	fitted model. Thereafter, you can use 'compare_fit()' to assess the comparative fit of the models. If the interaction effect makes the model better, and e.g., the RMSEA is good for the baseline model, the interaction model likely has a good RMSEA as well.
chisq	should Chi-Square based fit-measures be calculated?

get_pi_data

Get data with product indicators for different approaches

Description

get_pi_syntax is a function for creating the lavaan syntax used for estimating latent interaction models using one of the product indicators in lavaan.

Usage

```
get_pi_data(model.syntax, data, method = "dblcent", match = FALSE, ...)
```

Arguments

model.syntax	lavaan syntax
data	data to create product indicators from
method	method to use: "rca" = residual centering approach, "uca" = unconstrained approach, "dblcent" = double centering approach, "pind" = prod ind approach, with no constraints or centering, "custom" = use parameters specified in the function call
match	should product indicators be made using the match strategy
...	arguments passed to other functions (e.g., modsem_pi)

Value

data.frame

Examples

```
library(modsem)
library(lavaan)
m1 <- '
  # Outer Model
  X =~ x1 + x2 +x3
  Y =~ y1 + y2 + y3
  Z =~ z1 + z2 + z3

  # Inner model
  Y ~ X + Z + X:Z
  '
syntax <- get_pi_syntax(m1)
data <- get_pi_data(m1, oneInt)
est <- sem(syntax, data)
```

get_pi_syntax

Get lavaan syntax for product indicator approaches

Description

get_pi_syntax is a function for creating the lavaan syntax used for estimating latent interaction models using one of the product indicators in lavaan.

Usage

```
get_pi_syntax(model.syntax, method = "dblcent", match = FALSE, ...)
```

Arguments

<code>model.syntax</code>	lavaan syntax
<code>method</code>	method to use: "rca" = residual centering approach, "uca" = unconstrained approach, "dblcent" = double centering approach, "pind" = prod ind approach, with no constraints or centering, "custom" = use parameters specified in the function call
<code>match</code>	should product indicators be made using the match strategy
<code>...</code>	arguments passed to other functions (e.g., <code>modsem_pi</code>)

Value

character vector

Examples

```
library(modsem)
library(lavaan)
m1 <- '
  # Outer Model
  X =~ x1 + x2 +x3
  Y =~ y1 + y2 + y3
  Z =~ z1 + z2 + z3

  # Inner model
  Y ~ X + Z + X:Z
'
syntax <- get_pi_syntax(m1)
data <- get_pi_data(m1, oneInt)
est <- sem(syntax, data)
```

jordan

Jordan subset of PISA 2006 data

Description

The data stem from the large-scale assessment study PISA 2006 (Organisation for Economic Co-operation and Development, 2009) where competencies of 15-year-old students in reading, mathematics, and science are assessed using nationally representative samples in 3-year cycles. In this eacademicample, data from the student background questionnaire from the Jordan sample of PISA 2006 were used. Only data of students with complete responses to all 15 items (N = 6,038) were considered.

Format

A data frame of fifteen variables and 6,038 observations:

enjoy1 indicator for enjoyment of science, item ST16Q01: I generally have fun when I am learning <broad science> topics.

enjoy2 indicator for enjoyment of science, item ST16Q02: I like reading about <broad science>.

enjoy3 indicator for enjoyment of science, item ST16Q03: I am happy doing <broad science> problems.

enjoy4 indicator for enjoyment of science, item ST16Q04: I enjoy acquiring new knowledge in <broad science>.

enjoy5 indicator for enjoyment of science, item ST16Q05: I am interested in learning about <broad science>.

academic1 indicator for academic self-concept in science, item ST37Q01: I can easily understand new ideas in <school science>.

academic2 indicator for academic self-concept in science, item ST37Q02: Learning advanced <school science> topics would be easy for me.

academic3 indicator for academic self-concept in science, item ST37Q03: I can usually give good answers to <test questions> on <school science> topics.

academic4 indicator for academic self-concept in science, item ST37Q04: I learn <school science> topics quickly.

academic5 indicator for academic self-concept in science, item ST37Q05: <School science> topics are easy for me.

academic6 indicator for academic self-concept in science, item ST37Q06: When I am being taught <school science>, I can understand the concepts very well.

career1 indicator for career aspirations in science, item ST29Q01: I would like to work in a career involving <broad science>.

career2 indicator for career aspirations in science, item ST29Q02: I would like to study <broad science> after <secondary school>.

career3 indicator for career aspirations in science, item ST29Q03: I would like to spend my life doing advanced <broad science>.

career4 indicator for career aspirations in science, item ST29Q04: I would like to work on <broad science> projects as an adult.

Source

This version of the dataset, as well as the description was gathered from the documentation of the 'nlsem' package (<https://cran.r-project.org/package=nlsem>), where the only difference is that the names of the variables were changed

Originally the dataset was gathered by the Organisation for Economic Co-Operation and Development (2009). Pisa 2006: Science competencies for tomorrow's world (Tech. Rep.). Paris, France. Obtained from: <https://www.oecd.org/pisa/pisaproducts/database-pisa2006.htm>

Examples

```
## Not run:
m1 <- '
  ENJ =~ enjoy1 + enjoy2 + enjoy3 + enjoy4 + enjoy5
  CAREER =~ career1 + career2 + career3 + career4
  SC =~ academic1 + academic2 + academic3 + academic4 + academic5 + academic6
  CAREER ~ ENJ + SC + ENJ:ENJ + SC:SC + ENJ:SC
'

est <- modsem(m1, data = jordan)

## End(Not run)
```

 modsem

Interaction between latent variables

Description

modsem is a function for estimating interaction effects between latent variables, in structural equation models (SEM's). Methods for estimating interaction effects in SEM's can basically be split into two frameworks: 1. Product Indicator based approaches ("dblcent", "rca", "uca", "ca", "pind"), and 2. Distributionally based approaches ("lms", "qml"). For the product indicator based approaches, modsem() is essentially a just a fancy wrapper for lavaan::sem() which generates the necessary syntax, and variables for the estimation of models with latent product indicators. The distributionally based approaches are implemented in separately, and are not estimated using lavaan::sem(), but rather using custom functions (largely) written in C++ for performance reasons. For greater control, it is advised that you use one of the sub-functions (modsem_pi, modsem_da, modsem_mplus) directly, as passing additional arguments to them via modsem() can lead to unexpected behavior.

Usage

```
modsem(model.syntax = NULL, data = NULL, method = "dblcent", ...)
```

Arguments

model.syntax	lavaan syntax
data	dataframe
method	method to use: "rca" = residual centering approach (passed to lavaan), "uca" = unconstrained approach (passed to lavaan), "dblcent" = double centering approach (passed to lavaan), "pind" = prod ind approach, with no constraints or centering (passed to lavaan), "lms" = laten model structural equations (not passed to lavaan). "qml" = quasi maximum likelihood estimation of laten model structural equations (not passed to lavaan). "custom" = use parameters specified in the function call (passed to lavaan)
...	arguments passed to other functions depending on method (see modsem_pi, modsem_da, and modsem_mplus)

Value

modsem object

Examples

```

library(modsem)
# For more examples check README and/or GitHub.
# One interaction
m1 <- '
  # Outer Model
  X =~ x1 + x2 +x3
  Y =~ y1 + y2 + y3
  Z =~ z1 + z2 + z3

  # Inner model
  Y ~ X + Z + X:Z
  '

# Double centering approach
est1 <- modsem(m1, oneInt)
summary(est1)

## Not run:
# The Constrained Approach
est1_ca <- modsem(m1, oneInt, method = "ca")
summary(est1_ca)

# LMS approach
est1_lms <- modsem(m1, oneInt, method = "lms")
summary(est1_lms)

# QML approach
est1_qml <- modsem(m1, oneInt, method = "qml")
summary(est1_qml)

## End(Not run)

# Theory Of Planned Behavior
tpb <- '
# Outer Model (Based on Hagger et al., 2007)
ATT =~ att1 + att2 + att3 + att4 + att5
SN =~ sn1 + sn2
PBC =~ pbc1 + pbc2 + pbc3
INT =~ int1 + int2 + int3
BEH =~ b1 + b2

# Inner Model (Based on Steinmetz et al., 2011)
INT ~ ATT + SN + PBC
BEH ~ INT + PBC
BEH ~ INT:PBC
  '

```

```

# double centering approach
est_tpb <- modsem(tpb, data = TPB)
summary(est_tpb)

## Not run:
# The Constrained Approach
est_tpb_ca <- modsem(tpb, data = TPB, method = "ca")
summary(est_tpb_ca)

# LMS approach
est_tpb_lms <- modsem(tpb, data = TPB, method = "lms")
summary(est_tpb_lms)

# QML approach
est_tpb_qml <- modsem(tpb, data = TPB, method = "qml")
summary(est_tpb_qml)

## End(Not run)

```

modsemify

Generate parameter table for lavaan syntax

Description

Generate parameter table for lavaan syntax

Usage

```
modsemify(syntax)
```

Arguments

syntax model syntax

Value

data.frame with columns lhs, op, rhs, mod

Examples

```

library(modsem)
m1 <- '
# Outer Model
X =~ x1 + x2 +x3
Y =~ y1 + y2 + y3
Z =~ z1 + z2 + z3

# Inner model
Y ~ X + Z + X:Z

```

```
'
modsemify(m1)
```

 modsem_da

Interaction between latent variables using lms and qml approaches

Description

modsem_da is a function for estimating interaction effects between latent variables, in structural equation models (SEMs), using distributional analytic (DA) approaches. Methods for estimating interaction effects in SEM's can basically be split into two frameworks: 1. Product Indicator based approaches ("dblcent", "rca", "uca", "ca", "pind"), and 2. Distributionally based approaches ("lms", "qml"). modsem_da() handles the latter, and can estimate models using both qml and lms necessary syntax, and variables for the estimation of models with latent product indicators. NOTE: run 'default_settings_da()' to see default arguments.

Usage

```
modsem_da(
  model.syntax = NULL,
  data = NULL,
  method = "lms",
  verbose = NULL,
  optimize = NULL,
  nodes = NULL,
  convergence = NULL,
  optimizer = NULL,
  center.data = NULL,
  standardize.data = NULL,
  standardize.out = NULL,
  standardize = NULL,
  mean.observed = NULL,
  cov.syntax = NULL,
  double = NULL,
  calc.se = NULL,
  FIM = NULL,
  EFIM.S = NULL,
  OFIM.hessian = NULL,
  EFIM.parametric = NULL,
  robust.se = NULL,
  max.iter = NULL,
  max.step = NULL,
  fix.estep = NULL,
  start = NULL,
  epsilon = NULL,
  quad.range = NULL,
  ...
)
```

Arguments

<code>model.syntax</code>	lavaan syntax
<code>data</code>	dataframe
<code>method</code>	method to use: "lms" = laten model structural equations (not passed to lavaan). "qml" = quasi maximum likelihood estimation of laten model structural equations (not passed to lavaan).
<code>verbose</code>	should estimation progress be shown
<code>optimize</code>	should starting parameters be optimized
<code>nodes</code>	number of quadrature nodes (points of integration) used in lms, increased number gives better estimates but slower computation. How many is needed, depends on the complexity of the model For simple models you somewhere between 16-24 should be enough, for more complex higher numbers may be needed. For models where there is an interaction effects between and endogenous and exogenous variable the number of nodes should at least be 32, but practically (e.g., ordinal/skewed data) more than 32 is recommended. In cases, where data is non-normal it might be better to use the qml approach instead. For large numbers of nodes, you might want to change the 'quad.range' argument.
<code>convergence</code>	convergence criterion. Lower values give better estimates but slower computation.
<code>optimizer</code>	optimizer to use, can be either "nllminb" or "L-BFGS-B". For LMS, "nllminb" is recommended. For QML, "L-BFGS-B" may be faster if there is a large number of iterations, but slower if there are few iterations.
<code>center.data</code>	should data be centered before fitting model
<code>standardize.data</code>	should data be scaled before fitting model, will be overridden by standardize if standardize is set to TRUE. NOTE: It is recommended that you estimate the model normally and then standardize the output using 'standardized_estimates()'
<code>standardize.out</code>	should output be standardized (note will alter the relationships of parameter constraints, since to parameters are scaled unevenly, even if they have the same label). This does not alter the estimation of the model, only the output. NOTE: It is recommended that you estimate the model normally and then standardize the output using 'standardized_estimates()'
<code>standardize</code>	will standardize the data before fitting the model, remove the mean structure of the observed variables, and standardize the output. Note that standardize.data mean.observed, standardize.out will be overridden by standardize if standardize is set to TRUE. NOTE: It is recommended that you estimate the model normally and then standardize the output using 'standardized_estimates()'
<code>mean.observed</code>	should mean structure of the observed variables be estimated, will be overridden by standardize if standardize is set to TRUE. NOTE: Not recommended unless you know what you are doing.
<code>cov.syntax</code>	model syntax for implied covariance matrix (see 'vignette("interaction_two_etas", "modsem")')

double	try to double the number of dimensions of integrations used in LMS, this will be extremely slow, but should be more similar to mplus.
calc.se	should standard errors be computed, NOTE: If 'FALSE' information matrix will not be computed either
FIM	should fisher information matrix be calculated using observed or expected. must be either "observed" or "expected"
EFIM.S	if expected fisher information matrix is computed, EFIM.S selects the sample size of the generated data
OFIM.hessian	should observed fisher information be computed using hessian? if FALSE, it is computed using gradient
EFIM.parametric	should data for calculating expected fisher information matrix be simulated parametrically (simulated based on the assumptions- and implied parameters from the model), or non-parametrically (stochastically sampled). If you believe that normality assumptions are violated, 'EFIM.parametric = FALSE' might be the better option.
robust.se	should robust standard errors be computed? Meant to be used for QML, can be unreliable with the LMS-approach.
max.iter	max number of iterations
max.step	max steps for the M-step in the EM algorithm (LMS)
fix.estep	if TRUE, E-step will be fixed and the prior probabilities are set to the best prior probabilities, if loglikelihood is decreasing for more than 30 iterations.
start	starting parameters
epsilon	finite difference for numerical derivatives
quad.range	range in z-scores to perform numerical integration in LMS using Gaussian-Hermite Quadratures. By default Inf, such that f(t) is integrated from -Inf to Inf, but this will likely be inefficient and pointless at large number of nodes. Nodes outside +/- quad.range will be ignored.
...	additional arguments to be passed to the estimation function

Value

modsem_lms or modsem_qml object

Examples

```
library(modsem)
# For more examples check README and/or GitHub.
# One interaction
m1 <- "
# Outer Model
X =~ x1 + x2 + x3
Y =~ y1 + y2 + y3
Z =~ z1 + z2 + z3

# Inner model
```

```

    Y ~ X + Z + X:Z
  "

  ## Not run:
  # QML Approach
  est1 <- modsem_da(m1, oneInt, method = "qml")
  summary(est1)

  # Theory Of Planned Behavior
  tpb <- "
  # Outer Model (Based on Hagger et al., 2007)
  ATT =~ att1 + att2 + att3 + att4 + att5
  SN =~ sn1 + sn2
  PBC =~ pbc1 + pbc2 + pbc3
  INT =~ int1 + int2 + int3
  BEH =~ b1 + b2

  # Inner Model (Based on Steinmetz et al., 2011)
  # Covariances
  ATT ~~ SN + PBC
  PBC ~~ SN
  # Causal Relationships
  INT ~ ATT + SN + PBC
  BEH ~ INT + PBC
  BEH ~ INT:PBC
  "

  # lms approach
  estTpb <- modsem_da(tpb, data = TPB, method = lms)
  summary(estTpb)

  ## End(Not run)

```

 modsem_inspect

Inspect model information

Description

function used to inspect fitted object. similar to 'lavInspect()' argument 'what' decides what to inspect

Usage

```
modsem_inspect(object, what = NULL, ...)
```


Arguments

object	fitted model to inspect
what	what to inspect
...	Additional arguments passed to other functions

Details

for 'modsem_lms', 'modsem_qml' and 'modsem_lavaan' for 'modsem_lavaan', it is just a wrapper for 'lavInspect()' for 'modsem_lms' and 'modsem_qml' what can either be "all", "matrices", "optim", or just the name of what to extract.

modsem_mplus	<i>Estimation latent interactions through mplus</i>
--------------	---

Description

Estimation latent interactions through mplus

Usage

```
modsem_mplus(
  model.syntax,
  data,
  estimator = "ml",
  type = "random",
  algorithm = "integration",
  process = "8",
  ...
)
```

Arguments

model.syntax	lavaan/modsem syntax
data	dataset
estimator	estimator argument passed to mplus
type	type argument passed to mplus
algorithm	algorithm argument passed to mplus
process	process argument passed to mplus
...	arguments passed to other functions

Value

modsem_mplus object

Examples

```
# Theory Of Planned Behavior
tpb <- '
# Outer Model (Based on Hagger et al., 2007)
ATT =~ att1 + att2 + att3 + att4 + att5
SN =~ sn1 + sn2
PBC =~ pbc1 + pbc2 + pbc3
INT =~ int1 + int2 + int3
BEH =~ b1 + b2

# Inner Model (Based on Steinmetz et al., 2011)
# Covariances
ATT ~~ SN + PBC
PBC ~~ SN
# Causal Relationships
INT ~ ATT + SN + PBC
BEH ~ INT + PBC
BEH ~ INT:PBC
'

## Not run:
estTpbMplus <- modsem_mplus(tpb, data = TPB)
summary(estTpbLMS)

## End(Not run)
```

 modsem_pi

Interaction between latent variables using product indicators

Description

modsem_pi is a function for estimating interaction effects between latent variables, in structural equation models (SEMs), using product indicators. Methods for estimating interaction effects in SEM's can basically be split into two frameworks: 1. Product Indicator based approaches ("dblcent", "rca", "uca", "ca", "pind"), and 2. Distributionally based approaches ("lms", "qml"). modsem_pi() is essentially a just a fancy wrapper for lavaan::sem() which generates the necessary syntax, and variables for the estimation of models with latent product indicators. use 'default_settings_pi()' to get the default settings for the different methods.

Usage

```
modsem_pi(
  model.syntax = NULL,
  data = NULL,
  method = "dblcent",
  match = NULL,
  standardize.data = FALSE,
```

```

center.data = FALSE,
first.loading.fixed = TRUE,
center.before = NULL,
center.after = NULL,
residuals.prods = NULL,
residual.cov.syntax = NULL,
constrained.prod.mean = NULL,
constrained.loadings = NULL,
constrained.var = NULL,
constrained.res.cov.method = NULL,
auto.scale = "none",
auto.center = "none",
estimator = "ML",
group = NULL,
run = TRUE,
...
)

```

Arguments

<code>model.syntax</code>	lavaan syntax
<code>data</code>	dataframe
<code>method</code>	method to use: "rca" = residual centering approach (passed to lavaan), "uca" = unconstrained approach (passed to lavaan), "dblcent" = double centering approach (passed to lavaan), "pind" = prod ind approach, with no constraints or centering (passed to lavaan), "custom" = use parameters specified in the function call (passed to lavaan)
<code>match</code>	should the product indicators be created by using the match-strategy
<code>standardize.data</code>	should data be scaled before fitting model
<code>center.data</code>	should data be centered before fitting model
<code>first.loading.fixed</code>	Should the first factorloading in the latent prod be fixed to one?
<code>center.before</code>	should inds in prods be centered before computing prods (overwritten by method, if method != NULL)
<code>center.after</code>	should ind prods be centered after they have been computed?
<code>residuals.prods</code>	should ind prods be centered using residuals (overwritten by method, if method != NULL)
<code>residual.cov.syntax</code>	should syntax for residual covariances be produced (overwritten by method, if method != NULL)
<code>constrained.prod.mean</code>	should syntax prod mean be produced (overwritten by method, if method != NULL)

```

constrained.loadings      should syntax for constrained loadings be produced (overwritten by method, if
                          method != NULL)
constrained.var           should syntax for constrained variances be produced (overwritten by method, if
                          method != NULL)
constrained.res.cov.method  method for constraining residual covariances

auto.scale               methods which should be scaled automatically (usually not useful)
auto.center              methods which should be centered automatically (usually not useful)
estimator                estimator to use in lavaan
group                    group variable for multigroup analysis
run                       should the model be run via lavaan, if FALSE only modified syntax and data is
                          returned
...                       arguments passed to other functions, e.g., lavaan

```

Value

modsem object

Examples

```

library(modsem)
# For more examples check README and/or GitHub.
# One interaction
m1 <- '
  # Outer Model
  X =~ x1 + x2 +x3
  Y =~ y1 + y2 + y3
  Z =~ z1 + z2 + z3

  # Inner model
  Y ~ X + Z + X:Z
'

# Double centering approach
est1 <- modsem_pi(m1, oneInt)
summary(est1)

## Not run:
# The Constrained Approach
est1Constrained <- modsem_pi(m1, oneInt, method = "ca")
summary(est1Constrained)

## End(Not run)

# Theory Of Planned Behavior
tpb <- '
# Outer Model (Based on Hagger et al., 2007)

```

```
ATT =~ att1 + att2 + att3 + att4 + att5
SN =~ sn1 + sn2
PBC =~ pbc1 + pbc2 + pbc3
INT =~ int1 + int2 + int3
BEH =~ b1 + b2

# Inner Model (Based on Steinmetz et al., 2011)
# Covariances
ATT ~~ SN + PBC
PBC ~~ SN
# Causal Relationships
INT ~ ATT + SN + PBC
BEH ~ INT + PBC
BEH ~ INT:PBC
,

# double centering approach
estTpb <- modsem_pi(tpb, data = TPB)
summary(estTpb)

## Not run:
# The Constrained Approach
estTpbConstrained <- modsem_pi(tpb, data = TPB, method = "ca")
summary(estTpbConstrained)

## End(Not run)
```

multiplyIndicatorsCpp *Multiply indicators*

Description

Multiply indicators

Usage

```
multiplyIndicatorsCpp(df)
```

Arguments

df A data DataFrame

Value

A NumericVector

oneInt	<i>oneInt</i>
--------	---------------

Description

A simulated dataset with one interaction effect

parameter_estimates	<i>Extract parameterEstimates from an estimated model</i>
---------------------	---

Description

Extract parameterEstimates from an estimated model

Usage

```
parameter_estimates(object, ...)
```

Arguments

object	An object of class 'modsem_pi', 'modsem_da', or 'modsem_mplus'
...	Additional arguments passed to other functions

plot_interaction	<i>Plot Interaction Effects</i>
------------------	---------------------------------

Description

Plot Interaction Effects

Usage

```
plot_interaction(
  x,
  z,
  y,
  xz = NULL,
  vals_x = seq(-3, 3, 0.001),
  vals_z,
  model,
  alpha_se = 0.15,
  ...
)
```

Arguments

x	The name of the variable on the x-axis
z	The name of the moderator variable
y	The name of the outcome variable
xz	The name of the interaction term. If the interaction term is not specified, it will be created using 'x' and 'z'.
vals_x	The values of the x variable to plot, the more values the smoother the std.error-area will be
vals_z	The values of the moderator variable to plot. A separate regression line ("y ~ x z") will be plotted for each value of the moderator variable
model	An object of class 'modsem_pi', 'modsem_da', or 'modsem_mplus'
alpha_se	The alpha level for the std.error area
...	Additional arguments passed to other functions

Value

A ggplot object

Examples

```
library(modsem)
## Not run:
m1 <- "
# Outer Model
X =~ x1
X =~ x2 + x3
Z =~ z1 + z2 + z3
Y =~ y1 + y2 + y3

# Inner model
Y ~ X + Z + X:Z
"
est1 <- modsem(m1, data = oneInt)
plot_interaction("X", "Z", "Y", "X:Z", -3:3, c(-0.2, 0), est1)

tpb <- "
# Outer Model (Based on Hagger et al., 2007)
ATT =~ att1 + att2 + att3 + att4 + att5
SN =~ sn1 + sn2
PBC =~ pbc1 + pbc2 + pbc3
INT =~ int1 + int2 + int3
BEH =~ b1 + b2

# Inner Model (Based on Steinmetz et al., 2011)
# Causal Relationships
INT ~ ATT + SN + PBC
BEH ~ INT + PBC
# BEH ~ ATT:PBC
```

```

BEH ~ PBC:INT
# BEH ~ PBC:PBC
"

est2 <- modsem(tpb, TPB, method = "lms")
plot_interaction(x = "INT", z = "PBC", y = "BEH", xz = "PBC:INT",
                vals_z = c(-0.5, 0.5), model = est2)

## End(Not run)

```

standardized_estimates

Get standardized estimates

Description

Get standardized estimates

Usage

```
standardized_estimates(object, ...)
```

Arguments

object	An object of class ‘modsem_da’, ‘modsem_mplus’, or a parTable of class ‘data.frame’
...	Additional arguments passed to other functions

Details

for ‘modsem_lms’, ‘modsem_qml’ and ‘modsem_mplus’ objects, the interaction term is not standardized such that $\text{var}(xz) = 1$. The interaction term is not an actual variable in the model, meaning that it does not have a variance. It must therefore be calculated from the other parameters in the model. Assuming normality and zero-means the variance is calculated as ‘ $\text{var}(xz) = \text{var}(x) * \text{var}(z) + \text{cov}(x, z)^2$ ’. Thus setting the variance of the interaction term to 1, would only be ‘correct’ if the correlation between x and z is zero. This means that the standardized estimates for the interaction term will be different from those using lavaan, since there the interaction term is an actual latent variable in the model, with a standardized variance of 1.

summary.modsem_lms *summary for modsem objects*

Description

summary for modsem objects
summary for modsem objects
summary for modsem objects
summary for modsem objects

Usage

```
## S3 method for class 'modsem_lms'  
summary(  
  object,  
  H0 = TRUE,  
  verbose = TRUE,  
  r.squared = TRUE,  
  adjusted.stat = FALSE,  
  digits = 3,  
  scientific = FALSE,  
  ci = FALSE,  
  standardized = FALSE,  
  loadings = TRUE,  
  regressions = TRUE,  
  covariances = TRUE,  
  intercepts = TRUE,  
  variances = TRUE,  
  ...  
)
```

```
## S3 method for class 'modsem_qml'  
summary(  
  object,  
  H0 = TRUE,  
  verbose = TRUE,  
  r.squared = TRUE,  
  adjusted.stat = FALSE,  
  digits = 3,  
  scientific = FALSE,  
  ci = FALSE,  
  standardized = FALSE,  
  loadings = TRUE,  
  regressions = TRUE,  
  covariances = TRUE,  
  intercepts = TRUE,
```

```

    variances = TRUE,
    ...
)

## S3 method for class 'modsem_mplus'
summary(
  object,
  scientific = FALSE,
  standardize = FALSE,
  ci = FALSE,
  digits = 3,
  loadings = TRUE,
  regressions = TRUE,
  covariances = TRUE,
  intercepts = TRUE,
  variances = TRUE,
  ...
)

## S3 method for class 'modsem_pi'
summary(object, ...)

```

Arguments

object	modsem object to summarized
H0	should a null model be estimated (used for comparison)
verbose	print progress for the estimation of null model
r.squared	calculate R-squared
adjusted.stat	should sample size corrected/adjustes AIC and BIC be reported?
digits	number of digits to print
scientific	print p-values in scientific notation
ci	print confidence intervals
standardized	print standardized estimates
loadings	print loadings
regressions	print regressions
covariances	print covariances
intercepts	print intercepts
variances	print variances
...	arguments passed to lavaan::summary()
standardize	standardize estimates

Examples

```

## Not run:
m1 <- "
# Outer Model
X =~ x1 + x2 + x3
Y =~ y1 + y2 + y3
Z =~ z1 + z2 + z3

# Inner model
Y ~ X + Z + X:Z
"

est1 <- modsem(m1, oneInt, "lms")
summary(est1, ci = TRUE, scientific = TRUE)

## End(Not run)
## Not run:
m1 <- "
# Outer Model
X =~ x1 + x2 + x3
Y =~ y1 + y2 + y3
Z =~ z1 + z2 + z3

# Inner model
Y ~ X + Z + X:Z
"

est1 <- modsem(m1, oneInt, "qml")
summary(est1, ci = TRUE, scientific = TRUE)

## End(Not run)

```

TPB

TPB

Description

A simulated dataset based on the Theory of Planned Behaviour

Examples

```

tpb <- '
# Outer Model (Based on Hagger et al., 2007)
ATT =~ att1 + att2 + att3 + att4 + att5
SN =~ sn1 + sn2
PBC =~ pbc1 + pbc2 + pbc3
INT =~ int1 + int2 + int3
BEH =~ b1 + b2

# Inner Model (Based on Steinmetz et al., 2011)

```

```

INT ~ ATT + SN + PBC
BEH ~ INT + PBC + INT:PBC
,

est <- modsem(tpb, data = TPB)

```

TPB_UK

TPB_UK

Description

A dataset based on the Theory of Planned Behaviour from a UK sample. 4 variables with high communality were selected for each latent variable (ATT, SN, PBC, INT, BEH), from two time points (t1 and t2).

Source

Gathered from a replication study of the original by Hagger et al. (2023). Obtained from <https://doi.org/10.23668/psycharchives.2023.1>

Examples

```

tpb_uk <- '
# Outer Model (Based on Hagger et al., 2007)
ATT =~ att3 + att2 + att1 + att4
SN =~ sn4 + sn2 + sn3 + sn1
PBC =~ pbc2 + pbc1 + pbc3 + pbc4
INT =~ int2 + int1 + int3 + int4
BEH =~ beh3 + beh2 + beh1 + beh4

# Inner Model (Based on Steinmetz et al., 2011)
# Causal Relationships
INT ~ ATT + SN + PBC
BEH ~ INT + PBC
BEH ~ INT:PBC
,

est <- modsem(tpb_uk, data = TPB_UK)

```

trace_path

Estimate formulas for (co-)variance paths using Wright's path tracing rules

Description

This function estimates the path from x to y using the path tracing rules, note that it only works with structural parameters, so "=" are ignored. unless measurement.model = TRUE. you want to use the measurement model, "~" in the mod column of pt.

Usage

```
trace_path(
  pt,
  x,
  y,
  parenthesis = TRUE,
  measurement.model = FALSE,
  maxlen = 100,
  ...
)
```

Arguments

pt	A data frame with columns lhs, op, rhs, and mod, from modsemify(syntax)
x	source variable
y	destination variable
parenthesis	if TRUE, the output will be enclosed in parenthesis
measurement.model	if TRUE, the function will use the measurement model
maxlen	maximum length of a path before aborting
...	additional arguments passed to trace_path

Value

A string with the estimated path (simplified if possible)

Examples

```
library(modsem)
m1 <- '
# Outer Model
X =~ x1 + x2 +x3
Y =~ y1 + y2 + y3
Z =~ z1 + z2 + z3

# Inner model
Y ~ X + Z + X:Z
'
pt <- modsemify(m1)
trace_path(pt, "Y", "Y") # variance of Y
```

var_interactions	<i>Extract or modify parTable from an estimated model with estimated variances of interaction terms</i>
------------------	---

Description

Extract or modify parTable from an estimated model with estimated variances of interaction terms

Usage

```
var_interactions(object, ...)
```

Arguments

object	An object of class 'modsem_da', 'modsem_mplus', or a parTable of class 'data.frame'
...	Additional arguments passed to other functions

Index

[compare_fit](#), 3

[default_settings_da](#), 4
[default_settings_pi](#), 5

[extract_lavaan](#), 5

[fit_modsem_da](#), 6

[get_pi_data](#), 6
[get_pi_syntax](#), 7

[jordan](#), 8

[modsem](#), 10
[modsem_da](#), 13
[modsem_inspect](#), 16
[modsem_mplus](#), 17
[modsem_pi](#), 18
[modsemify](#), 12
[multiplyIndicatorsCpp](#), 21

[oneInt](#), 22

[parameter_estimates](#), 22
[plot_interaction](#), 22

[standardized_estimates](#), 24
[summary.modsem_lms](#), 25
[summary.modsem_mplus](#)
 ([summary.modsem_lms](#)), 25
[summary.modsem_pi](#) ([summary.modsem_lms](#)),
 25
[summary.modsem_qml](#)
 ([summary.modsem_lms](#)), 25

[TPB](#), 27
[TPB_UK](#), 28
[trace_path](#), 28

[var_interactions](#), 30