Package 'modsem'

December 16, 2024

Type Package

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

Version 1.0.5

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>.
``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>.
``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>.
``'lavaan': An R Package for Structural Equation Modeling."

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

LinkingTo Rcpp, RcppArmadillo

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

Depends R (>= 3.5.0)

URL https://modsem.org

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation yes

Author Kjell Solem Slupphaug [aut, cre] (<https://orcid.org/0009-0005-8324-2834>), Mehmet Mehmetoglu [ctb] (<https://orcid.org/0000-0002-6092-8551>), Matthias Mittner [ctb] (<https://orcid.org/0000-0003-0205-7353>)

Repository CRAN

Date/Publication 2024-12-16 14:10:01 UTC

Contents

`coef_modsem_da` *Wrapper for coef*

Description

wrapper for coef, to be used with modsem::coef_modsem_da, since coef is not in the namespace of modsem, but stats

Usage

```
coef_modsem_da(object, ...)
```

Arguments

`object`	fittet model to inspect
`...`	additional arguments

compare_fit *compare model fit for qml and lms models*

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_da‘ representing the null hypothesis model
estH1	object of class ‘modsem_da‘ 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

list

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

list

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 the product indicators be created by 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

`model.syntax`	`lavaan` syntax
`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 the product indicators be created by using the match-strategy
`...`	arguments passed to other functions (e.g., modsem_pi)

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` *Estimate interaction effects in structural equation models (SEMs)*

Description

`modsem()` is a function for estimating interaction effects between latent variables in structural equation models (SEMs). Methods for estimating interaction effects in SEMs can basically be split into two frameworks: 1. Product Indicator-based approaches (`"dblcent"`, `"rca"`, `"uca"`, `"ca"`, `"pind"`) 2. Distributionally based approaches (`"lms"`, `"qml"`).

For the product indicator-based approaches, `modsem()` is essentially 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 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" = latent model structural equations (not passed to lavaan), "qml" = quasi maximum likelihood estimation of latent 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 the method (see modsem_pi, modsem_da, and modsem_mplus)

Value

modsem object with class modsem_pi, modsem_da, or modsem_mplus

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", EFIM.S=1000)
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 SEMs can basically be split into two frameworks: 1. Product Indicator-based approaches (`"dblcent"`, `"rca"`, `"uca"`, `"ca"`, `"pind"`) 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,
  R.max = NULL,
  max.iter = NULL,
  max.step = NULL,
  fix.estep = NULL,
  start = NULL,
  epsilon = NULL,
  quad.range = NULL,
  n.threads = NULL,
  ...
)
```

Arguments

model.syntax	lavaan syntax
data	dataframe
method	method to use: "lms" = latent model structural equations (not passed to lavaan). "qml" = quasi maximum likelihood estimation of latent model structural equations (not passed to lavaan).
verbose	should estimation progress be shown
optimize	should starting parameters be optimized
nodes	number of quadrature nodes (points of integration) used in lms, increased number gives better estimates but slower computation. How many are needed depends on the complexity of the model. For simple models, somewhere between 16-24 nodes should be enough; for more complex models, higher numbers may be needed. For models where there is an interaction effect between an endogenous and exogenous variable, the number of nodes should be at least 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.
convergence	convergence criterion. Lower values give better estimates but slower computation.
optimizer	optimizer to use, can be either "nlminb" or "L-BFGS-B". For LMS, "nlminb" 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.
center.data	should data be centered before fitting model
standardize.data	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`.

`standardize.out`
: should output be standardized (note will alter the relationships of parameter constraints since 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`.

`standardize`
: 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`, and `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`.

`mean.observed`
: should the mean structure of the observed variables be estimated? This will be overridden by `standardize` if `standardize` is set to `TRUE`.

NOTE: Not recommended unless you know what you are doing.

`cov.syntax`
: model syntax for implied covariance matrix (see `vignette("interaction_two_etas", "modsem")`)

`double`
: try to double the number of dimensions of integration 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`, the information matrix will not be computed either.

`FIM`
: should the Fisher information matrix be calculated using the observed or expected values? Must be either `"observed"` or `"expected"`.

`EFIM.S`
: if the expected Fisher information matrix is computed, `EFIM.S` selects the number of Monte Carlo samples. Defaults to 100. **NOTE**: This number should likely be increased for better estimates (e.g., 1000-10000), but it might drasticly increase computation time.

`OFIM.hessian`
: should the observed Fisher information be computed using the Hessian? If `FALSE`, it is computed using the gradient.

`EFIM.parametric`
: should data for calculating the 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.

`R.max`
: Maximum population size (not sample size) used in the calculated of the expected fischer information matrix.

`max.iter`
: maximum number of iterations.

`max.step`
: maximum steps for the M-step in the EM algorithm (LMS).

fix.estep	if TRUE, the E-step will be fixed, and the prior probabilities will be set to the best prior probabilities, if the log-likelihood decreases 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 a large number of nodes. Nodes outside +/- quad.range will be ignored.
n.threads	number of cores to use for parallel processing. If NULL, it will use <= 2 threads. If an integer is specified, it will use that number of threads (e.g., n.threads = 4 will use 4 threads). If "default", it will use the default number of threads (2). If "max", it will use all available threads, "min" will use 1 thread.
...	additional arguments to be passed to the estimation function.

Value

modsem_da 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, EFIM.S = 1000)
summary(estTpb)

## End(Not run)
```

`modsem_inspect` *Inspect model information*

Description

function used to inspect fittet object. similar to ‘lavInspect()‘ argument ’what’ decides what to inspect

Usage

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

Arguments

`object`	fittet model to inspect
`what`	what to inspect
`...`	Additional arguments passed to other functions

Details

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

`modsem_mplus` *Estimation latent interactions through mplus*

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 Relationsships
  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 SEMs 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 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,
  na.rm = FALSE,
  suppress.warnings.lavaan = FALSE,
  suppress.warnings.match = FALSE,
  ...
)
```

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`), `"custom"` = use parameters specified in the function call (passed to `lavaan`)
`match`	should the product indicators be created by using the match-strategy
`standardize.data`	should data be scaled before fitting model
`center.data`	should data be centered before fitting model
`first.loading.fixed`	Should the first factor loading in the latent product be fixed to one?
`center.before`	should indicators in products be centered before computing products (overwritten by `method`, if `method != NULL`)
`center.after`	should indicator products be centered after they have been computed?
`residuals.prods`	should indicator products be centered using residuals (overwritten by `method`, if `method != NULL`)
`residual.cov.syntax`	should syntax for residual covariances be produced (overwritten by `method`, if `method != NULL`)
`constrained.prod.mean`	should syntax for product 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
na.rm	should missing values be removed (case-wise)? Defaults to FALSE. If TRUE, missing values are removed case-wise. If FALSE they are not removed.
suppress.warnings.lavaan	should warnings from lavaan be suppressed?
suppress.warnings.match	should warnings from match be suppressed?
...	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` *oneInt*

Description

A simulated dataset with one interaction effect

parameter_estimates *Extract parameterEstimates from an estimated model*

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 *Plot Interaction Effects*

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,
  digits = 2,
  ...
)
```

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. NOTE: vals_x are measured relative to the mean of x. The correct values will show up in the plot.
vals_z	The values of the moderator variable to plot. A separate regression NOTE: vals_z are measured relative to the mean of z. The correct values will show up in the plot. 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
digits	The number of digits to round the mean-shifted values of z
...	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 Relationsships
  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)
```

`plot_jn` *Plot Interaction Effect Using the Johnson-Neyman Technique*

Description

This function plots the simple slopes of an interaction effect across different values of a moderator variable using the Johnson-Neyman technique. It identifies regions where the effect of the predictor on the outcome is statistically significant.

Usage

```
plot_jn(
  x,
  z,
  y,
  xz = NULL,
  model,
  min_z = -3,
  max_z = 3,
  sig.level = 0.05,
  alpha = 0.2,
  detail = 1000,
  sd.line = 2,
  ...
)
```

Arguments

`x`	The name of the predictor variable (as a character string).
`z`	The name of the moderator variable (as a character string).
`y`	The name of the outcome variable (as a character string).
`xz`	The name of the interaction term. If not specified, it will be created using `x` and `z`.
`model`	A fitted model object of class `modsem_da`, `modsem_mplus`, `modsem_pi`, or `lavaan`.
`min_z`	The minimum value of the moderator variable `z` to be used in the plot (default is -3). It is relative to the mean of z.
`max_z`	The maximum value of the moderator variable `z` to be used in the plot (default is 3). It is relative to the mean of z.
`sig.level`	The significance level for the confidence intervals (default is 0.05).

alpha	alpha setting used in 'ggplot' (i.e., the opposite of opacity)
detail	The number of generated data points to use for the plot (default is 1000). You can increase this value for smoother plots.
sd.line	A thick black line showing +/- sd.line * sd(z). NOTE: This line will be truncated by min_z and max_z if the sd.line falls outside of [min_z, max_z].
...	Additional arguments (currently not used).

Details

The function calculates the simple slopes of the predictor variable x on the outcome variable y at different levels of the moderator variable z. It uses the Johnson-Neyman technique to identify the regions of z where the effect of x on y is statistically significant.

It extracts the necessary coefficients and variance-covariance information from the fitted model object. The function then computes the critical t-value and solves the quadratic equation derived from the t-statistic equation to find the Johnson-Neyman points.

The plot displays:

- The estimated simple slopes across the range of z.
- Confidence intervals around the slopes.
- Regions where the effect is significant (shaded areas).
- Vertical dashed lines indicating the Johnson-Neyman points.
- Text annotations providing the exact values of the Johnson-Neyman points.

Value

A ggplot object showing the interaction plot with regions of significance.

Examples

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

m1 <-  '
  visual  =~ x1 + x2 + x3
  textual =~ x4 + x5 + x6
  speed   =~ x7 + x8 + x9

  visual ~ speed + textual + speed:textual
'

est <- modsem(m1, data = lavaan::HolzingerSwineford1939, method = "ca")
plot_jn(x = "speed", z = "textual", y = "visual", model = est, max_z = 6)

## 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_da, and modsem_mplus objects, the interaction term is not standardized such that 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 var(xz) = var(x) * var(z) + 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_da *summary for modsem objects*

Description

summary for modsem objects

summary for modsem objects

summary for modsem objects

Usage

```
## S3 method for class 'modsem_da'
summary(
  object,
  H0 = TRUE,
  verbose = interactive(),
  r.squared = TRUE,
```

```
  adjusted.stat = FALSE,
  digits = 3,
  scientific = FALSE,
  ci = FALSE,
  standardized = FALSE,
  loadings = TRUE,
  regressions = TRUE,
  covariances = TRUE,
  intercepts = !standardized,
  variances = TRUE,
  var.interaction = FALSE,
  ...
)

## 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
var.interaction	if FALSE (default) variances for interaction terms will be removed (if present)
...	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, "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_1SO *TPB_1SO*

Description

A simulated dataset based on the Theory of Planned Behaviour, where INT is a higher order construct of ATT, SN, and PBC.

Examples

```
tpb <- '
  # First order constructs
  ATT =~ att1 + att2 + att3
  SN  =~ sn1 + sn2 + sn3
  PBC =~ pbc1 + pbc2 + pbc3
  BEH =~ b1 + b2

  # Higher order constructs
  INT =~ ATT + PBC + SN

  # Higher order interaction
  INTxPBC =~ ATT:PBC + SN:PBC + PBC:PBC

  # Structural model
  BEH ~ PBC + INT + INTxPBC
'

## Not run:
est <- modsem(tpb, data = TPB_2SO, method = "ca")
summary(est)

## End(Not run)
```

TPB_2SO *TPB_2SO*

Description

A simulated dataset based on the Theory of Planned Behaviour, where INT is a higher order construct of ATT and SN, and PBC is a higher order construct of PC and PB.

Examples

```
tpb <- "
  # First order constructs
  ATT =~ att1 + att2 + att3
  SN  =~ sn1 + sn2 + sn3
```

```
  PB =~ pb1 + pb2 + pb3
  PC =~ pc1 + pc2 + pc3
  BEH =~ b1 + b2

  # Higher order constructs
  INT =~ ATT + SN
  PBC =~ PC + PB

  # Higher order interaction
  INTxPBC =~ ATT:PC + ATT:PB + SN:PC + SN:PB

  # Structural model
  BEH ~ PBC + INT + INTxPBC
"

## Not run:
est <- modsem(tpb, data = TPB_2SO, method = "ca")
summary(est)

## End(Not run)
```

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 replciation study of the original by Hagger et al. (2023). Obtained from https://doi.org/10.23668/psycharchiv

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 Relationsships
 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`. If you want to use the measurement model, `"~"` should be in the `mod` column of `pt`.

Usage

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

Arguments

`pt`	A data frame with columns `lhs`, `op`, `rhs`, and `mod`, from modsemify
`x`	Source variable
`y`	Destination variable
`parenthesis`	If `TRUE`, the output will be enclosed in parenthesis
`missing.cov`	If `TRUE`, covariances missing from the model syntax will be added
`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, x = "Y", y = "Y", missing.cov = TRUE) # variance of Y
```

---

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

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

---

vcov_modsem_da *Wrapper for vcov*

Description

wrapper for vcov, to be used with modsem::vcov_modsem_da, since vcov is not in the namespace of modsem, but stats

Usage

```
vcov_modsem_da(object, ...)
```

Arguments

object	fittet model to inspect
...	additional arguments

Index