Package: missingmed (via r-universe)

June 12, 2026

Type Package

Title Mediation Analysis with Multiple Imputation for Missing Data

Version 0.2.0

Date 2026-06-11

Depends R (>= 4.1.0), methods (>= 4.1.0)

Imports S7 (>= 0.2.0), medfit (>= 0.3.1), RMediation (>= 1.4.0), mice (>= 3.0.0), lavaan (>= 0.6-0), OpenMx (>= 2.13), dplyr (>= 1.0.0), purrr (>= 0.3.0), tibble (>= 3.0.0), broom (>= 0.7.0), rlang (>= 0.4.0)

Suggests testthat (>= 3.0.0), knitr, rmarkdown, quarto, pkgdown

VignetteBuilder knitr

Remotes data-wise/medfit, RMediation=data-wise/rmediation

Additional_repositories https://data-wise.r-universe.dev

LazyData true

Description Provides S4 classes and methods for conducting mediation analysis with multiply imputed datasets. Integrates with the 'mice' package for multiple imputation and supports structural equation modeling (SEM) using either 'lavaan' or 'OpenMx'. Implements Rubin's rules for pooling parameter estimates and standard errors across imputations. Designed to work seamlessly with 'RMediation' for computing confidence intervals of indirect effects in the presence of missing data.

License GPL-2

Encoding UTF-8

Roxygen list(markdown = TRUE, roclets = c(``rd",``collate",``namespace"))

RoxygenNote 7.3.3

URL https://github.com/Data-Wise/missingmed, https://data-wise.github.io/missingmed/

BugReports https://github.com/Data-Wise/missingmed/issues

Repository https://data-wise.r-universe.dev

Date/Publication 2026-06-12 10:15:55 UTC

RemoteUrl https://github.com/Data-Wise/missingmed

RemoteRef dev

RemoteSha 3569e7ecdaae504b36fc2ba7ca3581e33e01bbd7

Contents

`fit_model` *Fit a Structural Equation Model*

Description

Fits a structural equation model to provided data using either the `lavaan` or `OpenMx` package.

Usage

```
fit_model(model, data)
```

Arguments

model	A character string representing lavaan model syntax, a lavaan model object, or an OpenMx::MxModel object.
data	A data frame containing the data to which the model will be fitted.

Value

A fitted model object from either lavaan or OpenMx, depending on the input.

Examples

```
## Not run:
data("HolzingerSwineford1939", package = "lavaan")
lav_model_syntax <- "visual =~ x1 + x2 + x3"
fitted_model <- fit_model(lav_model_syntax, HolzingerSwineford1939)

## End(Not run)
```

---

infer *Inference on the indirect effect under multiple imputation*

Description

Computes inference for the indirect (mediated) effect from a fitted missingmed pipeline, dispatching to one of two engines:

Usage

```
infer(object, ...)
```

Arguments

object	An MDMediationFit (supports both "mc" and "mbco") or an MDMediationResult (supports "mc").
...	Method arguments: type (inference type, "mc" (default) or "mbco"), level (confidence level for "mc", default 0.95), and n.mc (Monte-Carlo draws for "mc", default 1e5).

Details

- type = "mc" — Monte-Carlo / distribution-of-the-product confidence interval via RMediation::ci_mediation_data() applied to the **pooled** named medfit::MediationData.
- type = "mbco" — **D4-stacked MBCO** likelihood-ratio test of $H_0 : ab = 0$, computed from the per-imputation datasets (MBCO does not commute with Rubin's rules; see per_imputation_list()).

Value

For "mc", the list returned by RMediation::ci_mediation_data(). For "mbco", a named numeric vector c(D4, p, r4, nu, d_S).

See Also

run(), pool(), per_imputation_list()

---

is_fit *Determine If a SEM Model Has Been Fitted*

Description

This function checks whether a structural equation modeling (SEM) model, represented by either a lavaan object or an MxModel object from the OpenMx package, has been fitted. The function offers a convenient way to programmatically verify if the model fitting step has been executed for a given model object.

Method for ANY object. This method returns FALSE for any object that is not a lavaan or MxModel object.

Method for lavaan objects. This method checks the do.fit option in the lavaan object to determine if the model has been fitted. If the do.fit option is TRUE, the model has been fitted; otherwise, it has not been fitted.

Method for OpenMx objects. This method checks the wasRun slot in the MxModel object to determine if the model has been fitted. If the wasRun slot is TRUE, the model has been fitted; otherwise, it has not been fitted.

Usage

```
is_fit(model,...)

## S4 method for signature 'ANY'
is_fit(model, ...)

## S4 method for signature 'lavaan'
is_fit(model, ...)

## S4 method for signature 'MxModel'
is_fit(model, ...)
```

Arguments

model	A lavaan object.
...	Additional arguments affecting the method dispatched (currently not used but available for future extensions).

Value

A logical value: TRUE if the model has been fitted, and FALSE otherwise.

See Also

lavaan::lavaan, OpenMx::MxModel

Examples

```
## Not run:
# Assuming 'lav_model' is a lavaan model object
lav_model_fit <- is_fit(lav_model)

# Assuming 'mx_model' is an OpenMx model object
mx_model_fit <- is_fit(mx_model)

# Checking the output
print(lav_model_fit)
print(mx_model_fit)

## End(Not run)
```

---

is_lav_syntax *Function to check if lavaan model syntax is valid*

Description

This function checks if the lavaan model syntax is valid. It does so by attempting to parse and fit the model in a safe environment. If the model syntax is invalid, the function will return an error message.

Usage

```
is_lav_syntax(model, quiet = FALSE)
```

Arguments

model	A character string representing the lavaan model to be fitted.
quiet	A logical value indicating whether to suppress the error message. default is FALSE.

Value

A logical value indicating whether the model syntax is valid.

Author(s)

Davood Tofighi <dtofighi@gmail.com>

Examples

```
bad_model <- "y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9"
is_lav_syntax(bad_model)
good_model <- "visual =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed =~ x7 + x8 + x9
visual ~ speed
textual ~ speed"
is_lav_syntax(good_model)
```

---

is_pd *Checks if a matrix object is positive definite*

Description

Determines if a matrix is positive definite (all eigenvalues are strictly positive) by attempting Cholesky decomposition.

Usage

```
is_pd(x, quiet = FALSE)

## S4 method for signature 'matrix'
is_pd(x, quiet = FALSE)
```

Arguments

x	A numeric matrix.
quiet	Logical. If TRUE, suppresses warnings and error messages.

Value

Returns TRUE if the matrix is positive definite, FALSE otherwise.

Examples

```
# Example of a positive definite matrix
A <- matrix(c(1, 2, 2, 4), nrow = 2)
is_pd(A) # Should return TRUE
```

`is_valid_lav_syntax` *Function to check if lavaan model syntax is valid*

Description

This function checks if the lavaan model syntax is valid. It does so by attempting to parse and fit the model in a safe environment. If the model syntax is invalid, the function will return an error message.

Usage

```
is_valid_lav_syntax(model, data = NULL)
```

Arguments

`model`	A character string representing the lavaan model to be fitted.
`data`	A data frame containing the observed variables.

Value

A logical value indicating whether the model syntax is valid.

Author(s)

Davood Tofighi <dtofighi@gmail.com>

Examples

```
bad_model <- "y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9"
data(HolzingerSwineford1939, package = "lavaan")
is_valid_lav_syntax(bad_model, HolzingerSwineford1939)
good_model <- "visual =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed =~ x7 + x8 + x9
visual ~ speed
textual ~ speed"
is_valid_lav_syntax(good_model, HolzingerSwineford1939)
```

lav_mice *Fit SEM Model to Each Dataset in a MIDS Object Without Pooling*

Description

Fits a SEM model to each dataset in a `mids` object without pooling the results. This function is an extension for the `lavaan::sem()` function to handle mice::mids objects from the mice::mice package. It allows for both a SEM model syntax as a character string or a pre-fitted lavaan::lavaan model object.

Usage

```
lav_mice(model, mids, ...)
```

Arguments

`model`	Either a character string representing the SEM model to be fitted or a pre-fitted lavaan::lavaan model object.
`mids`	A `mids` object from the mice::mice package.
`...`	Additional arguments to be passed to `lavaan::sem()`.

Value

A list of lavaan::lavaan model fits, one for each imputed dataset.

Author(s)

Davood Tofighi <dtofighi@gmail.com>

Examples

```
## Not run:
# library(mice)
# library(lavaan)
# Load Holzinger and Swineford (1939) dataset
data("HolzingerSwineford1939", package = "lavaan")
# Introduce missing data
df_complete <- na.omit(HolzingerSwineford1939)
amp <- mice::ampute(df_complete, prop = 0.2, mech = "MAR")
data_with_missing <- amp$amp

# Perform multiple imputation
imputed_data <- mice::mice(data_with_missing, m = 3, maxit = 5, seed = 12345, printFlag = FALSE)

# fit the Holzinger and Swineford (1939) example model
HS_model <- " visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9 "
```

```
# Fit the SEM model without running
fit_HS <- lavaan::sem(HS_model, data = data_with_missing, do.fit = FALSE)
# Fit the SEM model without pooling to each imputed dataset
fit_list1 <- lav_mice(HS_model, imputed_data)
# 'fit_list1' now contains a list of lavaan objects, one for each imputed dataset
# Fit the SEM model without pooling to each imputed dataset using a pre-fitted model object
fit_list2 <- lav_mice(fit_HS, imputed_data)
# 'fit_list2' now contains a list of lavaan objects, one for each imputed dataset

## End(Not run)
```

`MDMediationData` *MDMediationData: imputed data + mediation specification (S7)*

Description

An S7 class holding multiply imputed data together with a **medfit-style mediation specification** (outcome/mediator formulas + roles). It is the entry point of the missingmed S7 pipeline (`set_md_mediation()` -> `run()` -> `pool()` -> `infer()`) and the S7 successor of the S4 SemImputedData class.

Usage

```
MDMediationData(
  data = NULL,
  formula_y = NULL,
  formula_m = NULL,
  treatment = character(0),
  mediator = character(0),
  engine = "glm",
  family_y = NULL,
  family_m = NULL,
  method = "mi",
  mechanism = "mar",
  weight_formula = NULL,
  weight_stabilize = TRUE,
  weight_trim = 1,
  se_type = "sandwich",
  conf_int = FALSE,
  conf_level = 0.95,
  n_imputations = integer(0),
  original_data = data.frame()
)
```

Arguments

`data`	For `method = "mi"`, an object of class `mids` from the `mice` package; for `method = "ipw"`, a raw `data.frame` (the complete cases are reweighted).

`formula_y` Outcome model formula (e.g. `Y ~ X + M + C`).

`formula_m` Mediator model formula (e.g. `M ~ X + C`).

`treatment` Name of the treatment/exposure variable.

`mediator` Name of the mediator variable.

`engine` medfit fitting engine, e.g. `"glm"` (default).

`family_y, family_m` `stats::family` objects for the outcome and mediator models. Default `stats::gaussian()`.

`method` Estimator axis: `"mi"` (default) or `"ipw"`.

`mechanism` Assumed missing-data mechanism: `"mar"` (default) or `"mnar"`.

`weight_formula` (IPW) Missingness model specification: `NULL` (default; use all observed predictors), a single `formula` (joint complete-case model), or a named `list` of formulas (per-variable models). Ignored for MI.

`weight_stabilize` (IPW) Logical; if `TRUE` (default) use stabilized weights `P(R=1|X) / P(R=1|Z)`. Ignored for MI.

`weight_trim` (IPW) Upper quantile at which to cap weights (e.g. `0.99`); `1` (default) disables trimming. Ignored for MI.

`se_type` (IPW) Variance estimator passed to `medfit::fit_mediation()`: `"sandwich"` (default for IPW, HC robust) or `"model"`. Ignored for MI.

`conf_int` Logical; whether downstream output carries confidence intervals. Defaults to `FALSE`.

`conf_level` Numeric in (0, 1); confidence level. Defaults to `0.95`.

`n_imputations` Number of imputations (MI) or `1` (IPW).

`original_data` The original data (pre-imputation for MI; the supplied frame for IPW).

Details

Fitting is delegated to `medfit::fit_mediation()` (one call per imputation), so the per-imputation fits carry **named** path coefficients (`a`, `b`, `c_prime`) ready for RMediation::RMediation inference. The estimator (`method`) and the model are orthogonal axes: `method` selects the missing-data estimator (`"mi"`/`"ipw"`); the formulas/engine select the model.

Value

An `MDMediationData` S7 object.

See Also

`set_md_mediation()`, `medfit::fit_mediation()`, SemImputedData

`MDMediationFit` *MDMediationFit: per-imputation mediation fits (S7)*

Description

An S7 class holding the result of fitting a mediation model across all imputations. Its defining feature is `per_imputation`: a list of **named** medfit::MediationData objects, one per imputation. This list is what the MBCO-MI path consumes, because MBCO does not commute with Rubin's rules (D4-stacked MBCO needs the per-imputation fits, not the pooled estimate).

Usage

```
MDMediationFit(
  per_imputation = list(),
  fits = list(),
  m = integer(0),
  engine = "glm",
  conf_int = FALSE,
  conf_level = 0.95,
  weights = NULL,
  source = NULL
)
```

Arguments

`per_imputation`	A list of named medfit::MediationData objects (length `m`).
`fits`	A list of the raw backend fits (`lavaan`/`OpenMx`), one per imputation.
`m`	Integer number of imputations.
`engine`	medfit fitting engine used (e.g. `"glm"`).
`conf_int`	Logical; whether output carries confidence intervals.
`conf_level`	Numeric in (0, 1); confidence level.
`weights`	(IPW) Full-length numeric IPW weight vector (`NA` for dropped rows); `NULL` for MI fits.
`source`	The originating MDMediationData (retained so MBCO can refit constrained/unconstrained models against the imputed data).

Details

It is the S7 successor of the S4 SemResults class.

Value

An `MDMediationFit` S7 object.

See Also

`run()`, `per_imputation_list()`, SemResults

`MDMediationResult` *MDMediationResult: pooled mediation result (S7)*

Description

An S7 class holding the Rubin's-rules pooled mediation result. Its defining feature is `pooled`: a single **named** medfit::MediationData built from the pooled estimates and total variance-covariance, valid as input to `RMediation::ci_mediation_data()` / `RMediation::medci()` (path coefficients resolve by name). It is the S7 successor of the S4 PooledSEMResults class.

Usage

```
MDMediationResult(
  pooled = NULL,
  tidy_table = data.frame(),
  cov_total = NULL,
  cov_between = NULL,
  cov_within = NULL,
  m = integer(0),
  engine = "glm",
  conf_int = FALSE,
  conf_level = 0.95
)
```

Arguments

`pooled`	A named medfit::MediationData carrying the pooled estimates and total vcov (path labels `a`, `b`, `c_prime`, ...).
`tidy_table`	A data frame of pooled estimates (term, estimate, std_error, p_value, var_w, var_b, var_tot).
`cov_total`, `cov_between`, `cov_within`	The Rubin's-rules total, between-, and within-imputation covariance matrices.
`m`	Integer number of imputations pooled.
`engine`	medfit fitting engine used (e.g. `"glm"`).
`conf_int`	Logical; whether the tidy table carries confidence intervals.
`conf_level`	Numeric in (0, 1); confidence level.

Value

An `MDMediationResult` S7 object.

See Also

`pool()`, `infer()`, PooledSEMResults

mx_mice *Fit OpenMx model to multiply imputed datasets*

Description

This function fits an OpenMx model to each imputed dataset in a 'mids' object from the 'mice' package. The function returns a list of OpenMx model fits.

Usage

```
mx_mice(model, mids, ...)
```

Arguments

model	An OpenMx model object.
mids	A 'mids' object from the 'mice' package.
...	Additional arguments to be passed to 'mxRun'.

Value

A list of OpenMx model fits.

Author(s)

Davood Tofighi <dtofighi@gmail.com>

Examples

```
## Not run:
# library(OpenMx)
# library(mice)
# Fit a model to multiply imputed datasets
data("HolzingerSwineford1939", package = "lavaan")
# Introduce missing data
df_complete <- na.omit(HolzingerSwineford1939)
amp <- mice::ampute(df_complete, prop = 0.2, mech = "MAR")
df_incomplete <- amp$amp
# Perform multiple imputation
imputed_data <- mice(df_incomplete, m = 3, method = "pmm", maxit = 5, seed = 12345)
# Simple SEM model specification with OpenMx
manifestVars <- paste0("x", 1:9)
latVar <- c("visual", "textual", "speed")
model <- mxModel("Simple SEM",
  type = "RAM",
  manifestVars = manifestVars,
  latentVars = latVar,
  mxPath(from = "visual", to = c("x1", "x2", "x3")),
  mxPath(from = "textual", to = c("x4", "x5", "x6")),
  mxPath(from = "speed", to = c("x7", "x8", "x9")),
```

```
  mxPath(from = manifestVars, arrows = 2),
  mxPath(from = latVar, arrows = 2, free = FALSE, values = 1.0),
  mxPath(from = "one", to = manifestVars, arrows = 1, free = TRUE, values = 1.0),
  mxPath(from = "one", to = latVar, arrows = 1, free = FALSE, values = 0),
  mxData(df_complete, type = "raw")
)
# Assuming mx_mice is correctly defined in your environment
fits <- mx_mice(model, imputed_data)
summary(fits[[1]])

## End(Not run)
```

`n_imp` *Get Number of Imputations from a mids Object*

Description

This function returns the number of imputations stored in a `mids` object created by the `mice` package.

Usage

```
n_imp(x)

## S4 method for signature 'mids'
n_imp(x)
```

Arguments

`x`	A `mids` object representing multiple imputed datasets.

Value

An integer representing the number of imputations.

Examples

```
## Not run:
# Assuming `imputed_data` is a mids object created by the mice package
n_imp(imputed_data)

## End(Not run)
```

n_imputations *Number of imputations*

Description

Number of imputations

Usage

```
n_imputations(object, ...)
```

Arguments

object	An MDMediationFit or MDMediationResult object.
...	Unused.

Value

Integer count of imputations.

per_imputation_list *Access the per-imputation mediation fits (for MBCO)*

Description

Returns the list of per-imputation **named** medfit::MediationData objects held in an MDMediationFit, together with the number of imputations `m`.

Usage

```
per_imputation_list(object, ...)
```

Arguments

object	An MDMediationFit object.
...	Unused.

Details

This accessor exists because **MBCO does not commute with Rubin's rules**: D4-stacked MBCO needs the per-imputation fits, not the pooled estimate. The list it returns is the shape consumed by `infer()(type = "mbco")` and by an external `RMediation::mbco()` MI entry point (missingmed issue #2).

Value

A list with components `per_imputation` (a length-`m` list of named medfit::MediationData) and `m` (the number of imputations).

See Also

`run()`, `infer()`

`pool` *Pool per-imputation mediation fits with Rubin's rules*

Description

Applies Rubin's (1987) rules to the list of per-imputation **named** medfit::MediationData objects in an MDMediationFit, producing a single pooled named medfit::MediationData (the `pooled` slot of the returned MDMediationResult). Because the estimates and variance-covariance carry the mediation path names (`a`, `b`, `c_prime`, ...), the pooled object is valid input to `RMediation::ci_mediation_data()` / `RMediation::medci()`.

Usage

```
pool(object, ...)
```

Arguments

`object`	An MDMediationFit object.
`...`	Unused.

Details

Pooling math (migrated from the S4 `pool_sem` / `pool_tidy` / `pool_cov`):

$$\bar{Q} = \frac{1}{m}\sum_i Q_i, \quad \bar{U} = \frac{1}{m}\sum_i U_i, \quad B = \mathrm{cov}(Q_1, \ldots, Q_m), \quad T = \bar{U} + (1 + 1/m)B.$$

It is the S7 successor of the S4 `pool_sem()` method.

Value

An MDMediationResult object.

References

Rubin, D. B. (1987). *Multiple Imputation for Nonresponse in Surveys*. Wiley.

See Also

`run()`, `infer()`, `pool_sem()`

pool_sem *Pool SEM Analysis Results*

Description

pool_sem pools SEM analysis results, supporting lavaan and OpenMx models. It calculates pooled estimates, standard errors, confidence intervals, and more.

This function pools the results of structural equation modeling (SEM) analyses performed on multiple imputed datasets. It supports pooling for models analyzed with either the lavaan or OpenMx package. The function extracts and pools relevant statistics (e.g., estimates, standard errors) across all imputations, considering the specified confidence interval settings.

Usage

```
pool_sem(object)

## S4 method for signature 'SemResults'
pool_sem(object)
```

Arguments

object	SemResults object with SEM analysis results.

Details

A generic function to pool SEM analysis results from multiple datasets or imputations.

Refer to method-specific documentation for details on pooling process and assumptions.

Value

PooledSEMResults object containing pooled SEM analysis results.

A data.frame containing the pooled results of the SEM analyses. The column names adhere to tidy conventions and include the following columns:

- term: The name of the parameter being estimated.
- estimate: The pooled estimate of the parameter.
- std_error: The pooled standard error of the estimate.
- statistic: The pooled test statistic (e.g., z-value, t-value).
- p_value: The pooled p-value for the test statistic.
- conf_low: The lower bound of the confidence interval for the estimate.
- conf_high: The upper bound of the confidence interval for the estimate.

Author(s)

Davood Tofighi <dtofighi@gmail.com>

See Also

lavaan::lavaan, OpenMx::OpenMx

Examples

```
## Not run:
# Assuming `sem_results` is a SemResults object with lavaan model fits:
library(RMediation)
# Load Holzinger and Swineford (1939) dataset
data("HolzingerSwineford1939", package = "lavaan")
# Introduce missing data
df_complete <- na.omit(HolzingerSwineford1939[paste0("x", 1:9)])
amp <- mice::ampute(df_complete, prop = 0.1, mech = "MAR")
data_with_missing <- amp$amp
# Perform multiple imputation
imputed_data <- mice::mice(data_with_missing, m = 3, maxit = 3, seed = 12345, printFlag = FALSE)
model <- "
 visual  =~ x1 + x2 + x3
 textual =~ x4 + x5 + x6
 speed   =~ x7 + x8 + x9
 "
res_pooled <- imputed_data |>
  set_sem(model) |>
  run_sem() |>
  pool_sem()
res_pooled@tidy_table # Print the pooled results

## End(Not run)
```

PooledSEMResults *Pooled SEM Analysis Results Class*

Description

An S4 class to represent pooled results from SEM analysis across multiple imputations or datasets. It contains pooled estimates, standard errors, test statistics, p-values, and confidence intervals for each parameter estimated across multiple imputations.

Slots

`tidy_table` data.frame A data frame containing the pooled results of the SEM analyses. The column names adhere to tidy conventions and include the following columns:

- `term`: The name of the parameter being estimated.
- `estimate`: The pooled estimate of the parameter.
- `std_error`: The pooled standard error of the estimate.
- `p_value`: The pooled p-value for the test statistic.
- `conf_low`: The lower bound of the confidence interval for the estimate.

- conf_high: The upper bound of the confidence interval for the estimate.

cov_total matrix The pooled total covariance matrix of the parameter estimates.

cov_between matrix The pooled between-imputation covariance matrix of the parameter estimates.

cov_within matrix The pooled within-imputation covariance matrix of the parameter estimates.

method character The method used for SEM analysis ('lavaan' or 'OpenMx').

conf_int logical Whether to calculate confidence intervals for the pooled estimates. default is FALSE.

conf_level numeric The confidence level used in the interval calculation. default is 0.95.

Author(s)

Davood Tofighi <dtofighi@gmail.com>

---

run *Fit the mediation model across imputations*

Description

Runs the mediation specification held in an MDMediationData object on every imputed dataset, delegating each fit to `medfit::fit_mediation()`. The result is an MDMediationFit whose `per_imputation` slot is a list of **named** medfit::MediationData objects (one per imputation) — the shape consumed by both Rubin's-rules pooling (`pool()`) and D4-stacked MBCO (`infer()`).

Usage

```
run(object, ...)
```

Arguments

object	An MDMediationData object.
...	Additional arguments forwarded to `medfit::fit_mediation()`.

Details

It is the S7 successor of the S4 `run_sem()` method.

Value

An MDMediationFit object.

See Also

`set_md_mediation()`, `pool()`, `infer()`, `run_sem()`

run_sem *Run a SEM model*

Description

A generic function to run and analyze multiply imputed data sets.

This method facilitates running SEM analysis using either lavaan or OpenMx on multiply imputed datasets contained within a SemImputedData object.

Usage

```
run_sem(object, ...)

## S4 method for signature 'SemImputedData'
run_sem(object, ...)
```

Arguments

object	A SemImputedData object
...	Additional arguments passed to either lavaan::sem or OpenMx::MxModel.

Value

A SemResults object

Author(s)

Davood Tofighi <dtofighi@gmail.com>

Examples

```
## Not run:
library(RMediation)
# Load Holzinger and Swineford (1939) dataset
data("HolzingerSwineford1939", package = "lavaan")
# Introduce missing data
df_complete <- na.omit(HolzingerSwineford1939[paste0("x", 1:9)])
amp <- mice::ampute(df_complete, prop = 0.1, mech = "MAR")
data_with_missing <- amp$amp
# Perform multiple imputation
imputed_data <- mice::mice(data_with_missing, m = 3, maxit = 3, seed = 12345, printFlag = FALSE)
model <- "
 visual  =~ x1 + x2 + x3
 textual =~ x4 + x5 + x6
 speed   =~ x7 + x8 + x9
 "
res_sem <- imputed_data |>
  set_sem(model) |>
```

```
  run_sem()
res_sem@estimate_df # long tidy table of estimates across imputed datasets

## End(Not run)
```

SemImputedData *SemImputedData Class*

Description

An S4 class to hold multiply imputed datasets for structural equation modeling (SEM) analysis. It facilitates working with imputed data from the `mice` package and supports SEM analysis using either the `lavaan` or `OpenMx` packages.

Slots

`data` An object of class `mids` from the `mice` package, representing multiply imputed datasets.

`model` A `lavaan` or `OpenMx` model syntax to be used for SEM analysis. For `lavaan` models, the syntax should be a character string as described in lavaan::model.syntax. For `OpenMx` models, the syntax should be an OpenMx::mxModel object with or without `OpenMx::mxData()` specified; that is, `mxModel` syntax can be without data specified. In addition, both `lavaan` and `OpenMx` models can be a fitted model object in the respective package.

`method` A character string indicating the SEM package to be used for analysis. It is a derived slot from the `model` slot, and it is set automatically based on the class of the `model` slot. The possible values are "lavaan" or "OpenMx".

`conf_int` A logical value indicating whether confidence intervals are included in the SEM results. Defaults to `FALSE`.

`conf_level` A numeric value specifying the confidence level for confidence intervals, which must be between 0 and 1. Defaults to 0.95.

`original_data` A derived (from mids object) slot to store the original data used to create the imputed datasets.

`n_imputations` A derived (from mids object) slot to store the number of imputations used to create the imputed datasets.

`fit_model` SEM fitted to the original data with list wise deletion of missing data.

Author(s)

Davood Tofighi <dtofighi@gmail.com>

SemResults *SemResults Class*

Description

An S4 class for storing the results of SEM analysis performed on multiply imputed datasets. Supports `lavaan` and `OpenMx`.

Slots

`results` A list of SEM model fits for each imputed dataset.

`estimate_df` Data frame of parameter estimates and standard errors.

`coef_df` Data frame of coefficient estimates for each imputed dataset.

`cov_df` List of covariance matrices of coefficient estimates.

`method` SEM package used for analysis: 'lavaan' or 'OpenMx'.

`conf_int` Logical; if confidence intervals are included.

`conf_level` Confidence level for confidence intervals.

Author(s)

Davood Tofighi <dtofighi@gmail.com>

set_md_mediation *Set up a mediation analysis with missing data (MI or IPW)*

Description

Constructs an MDMediationData object: the entry point of the missingmed S7 pipeline. It records a **medfit-style mediation specification** (outcome and mediator formulas plus the treatment/mediator roles) together with the data. Fitting is delegated to `medfit::fit_mediation()` downstream by `run()`. It is the S7 successor of the S4 `set_sem()` constructor.

Usage

```
set_md_mediation(
  data,
  formula_y,
  formula_m,
  treatment,
  mediator,
  engine = "glm",
  family_y = stats::gaussian(),
  family_m = stats::gaussian(),
  method = c("mi", "ipw"),
```

```
  mechanism = c("mar", "mnar"),
  weight_formula = NULL,
  weight_stabilize = TRUE,
  weight_trim = 1,
  se_type = c("sandwich", "model"),
  conf_int = FALSE,
  conf_level = 0.95
)
```

Arguments

data	For method = "mi", a mice::mids object; for method = "ipw", a data.frame (may contain NAs; complete cases are reweighted).
formula_y	Outcome model formula (e.g. Y ~ X + M + C).
formula_m	Mediator model formula (e.g. M ~ X + C).
treatment	Name of the treatment/exposure variable.
mediator	Name of the mediator variable.
engine	medfit fitting engine. Defaults to "glm".
family_y, family_m	stats::family objects for the outcome and mediator models. Default stats::gaussian().
method	Estimator axis: "mi" (default) or "ipw".
mechanism	Assumed missing-data mechanism: "mar" (default) or "mnar".
weight_formula	(IPW) Missingness model: NULL (default; all observed predictors), a single formula, or a named list of per-variable formulas.
weight_stabilize	(IPW) Use stabilized weights? Default TRUE.
weight_trim	(IPW) Upper quantile to cap weights; 1 (default) = none.
se_type	(IPW) "sandwich" (default, HC robust) or "model".
conf_int	Logical; whether downstream output carries confidence intervals. Defaults to FALSE.
conf_level	Numeric in (0, 1); confidence level. Defaults to 0.95.

Details

Two estimators share the interface (method):

- "mi" — data is a mice::mids object; run() fits every imputation.
- "ipw" — data is a raw data.frame; run() reweights the complete cases by inverse missingness probability and fits once.

Value

An MDMediationData object.

See Also

MDMediationData, `run()`, `pool()`, `infer()`, `medfit::fit_mediation()`

Examples

```
## Not run:
set.seed(1)
d <- data.frame(X = rbinom(200, 1, .5), C = rnorm(200))
d$M <- .5 * d$X + .3 * d$C + rnorm(200)
d$Y <- .2 * d$X + .4 * d$M + .3 * d$C + rnorm(200)
d$M[sample(200, 30)] <- NA
# MI
imp <- mice::mice(d, m = 5, printFlag = FALSE)
md_mi <- set_md_mediation(imp, Y ~ X + M + C, M ~ X + C,
  treatment = "X", mediator = "M")
# IPW (raw data.frame)
md_ipw <- set_md_mediation(d, Y ~ X + M + C, M ~ X + C,
  treatment = "X", mediator = "M", method = "ipw")

## End(Not run)
```

`set_sem` *Set up an SEM model with multiply imputed data.*

Description

This function sets up an SEM model with multiply imputed data for analysis. The function accepts a mice::mids object and a model syntax for either lavaan::lavaan or OpenMx::OpenMx and returns a SemImputedData object for analysis. It returns an error if the provided data is not a mice::mids object or if the specified SEM analysis method is not supported. It returns an object of class SemImputedData.

Usage

```
set_sem(data, model, conf_int = FALSE, conf_level = 0.95)

## S4 method for signature 'mids'
set_sem(data, model, conf_int = FALSE, conf_level = 0.95)
```

Arguments

`data`	A mice::mids object from the `mice` package containing multiply imputed datasets.
`model`	A lavaan::lavaan or OpenMx::OpenMx model syntax to be used for SEM analysis. For `lavaan` models, the syntax should be a character string as described in lavaan::model.syntax. For `OpenMx` models, the syntax should be an OpenMx::mxModel object with or without `OpenMx::mxData()` specified; that is, `mxModel` syntax can be without data specified. In addition, both `lavaan` and `OpenMx` models can be a fitted model object in the respective package.

conf_int	A logical value indicating whether confidence intervals are included in the SEM results. Defaults to FALSE.
conf_level	A numeric value specifying the confidence level for confidence intervals, which must be between 0 and 1. Defaults to 0.95. If conf_int is FALSE, this argument is ignored.

Details

The function technically constructs a new SemImputedData object for structural equation modeling (SEM) analysis using either lavaan::lavaan or OpenMx::OpenMx on multiply imputed datasets. This function ensures that the provided data is a mice::mids object from the mice package and that the specified SEM analysis method is supported.

All the arguments data, method, conf_int, and conf_level are used to specify the SEM analysis. set_sem is a constructor function for SemImputedData class. These methods are used as constructors for the SemImputedData class.

Value

An object of class SemImputedData. See SemImputedData for the details of the slots.

See Also

SemImputedData mice::mids lavaan::lavaan OpenMx::OpenMx

Examples

```
## Not run:
data("HolzingerSwineford1939", package = "lavaan")
df_complete <- na.omit(HolzingerSwineford1939)
amp <- mice::ampute(df_complete, prop = 0.2, mech = "MAR")
imputed_data <- mice::mice(amp$amp, m = 3, maxit = 3, seed = 12345, printFlag = FALSE)
model <- "
visual  =~ x1 + x2 + x3
textual =~ x4 + x5 + x6
speed   =~ x7 + x8 + x9"
sem_data <- set_sem(imputed_data, model)
str(sem_data)

## End(Not run)
```

---

show,SemImputedData-method

Show SemImputedData

Description

Method to display a concise summary of a SemImputedData object. This method is automatically called when you print a SemImputedData object.

Usage

```
## S4 method for signature 'SemImputedData'
show(object)
```

Arguments

object	The SemImputedData object to be displayed.

Value

This method does not return a value but displays a summary of the SemImputedData object.

Examples

```
## Not run:
data("HolzingerSwineford1939", package = "lavaan")
hs_data <- HolzingerSwineford1939[paste0("x", 1:9)] |> mice::ampute()
hs_data <- hs_data$amp
imp_data <- mice::mice(HolzingerSwineford1939, m = 5)
model_lav <- "visual  =~ x1 + x2 + x3
              textual =~ x4 + x5 + x6
              speed   =~ x7 + x8 + x9"
imp_data <- mice::mice(hs_data, m = 5)
sem_data <- SemImputedData(imp_data, model_lav)
show(sem_data)

## End(Not run)
```

---

summary,SemImputedData-method

Summary Method for SemImputedData Objects

Description

Provides a comprehensive summary of an SemImputedData object, including the SEM method used, the number of imputations, basic information about the imputed data, and summaries of the original data and fitted model.

Usage

```
## S4 method for signature 'SemImputedData'
summary(object, ...)
```

Arguments

object	An object of class SemImputedData.
...	Further arguments passed to or from other methods.

Value

A textual summary of the SemImputedData object.

Examples

```
## Not run:
# Assuming `sem_imputed_data` is an SemImputedData object
summary(sem_imputed_data)

## End(Not run)
```

`tidy.logLik` *Creates a data.frame for a log-likelihood object*

Description

Creates a data.frame for a log-likelihood object

Usage

```
## S3 method for class 'logLik'
tidy(x, ...)
```

Arguments

`x`	A log-likelihood object, typically returned by logLik.
`...`	Additional arguments (not used)

Value

A data.frame with columns:

term The term name

estimate The log-likelihood value

df The degrees of freedom

Author(s)

Davood Tofighi <dtofighi@gmail.com>

See Also

logLik

Examples

```
fit <- lm(mpg ~ wt, data = mtcars)
logLik_fit <- logLik(fit)
# Dispatch via the broom generic (registered against broom::tidy).
tidy(logLik_fit)
```

`tidy.MxModel` *Tidy an MxModel Object*

Description

Extracts parameter estimates from an OpenMx::MxModel from the OpenMx::OpenMx model and formats them into a tidy dataframe.

Usage

```
## S3 method for class 'MxModel'
tidy(x, conf_int = FALSE, conf_level = 0.95, ...)
```

Arguments

`x`	An object of class OpenMx::MxModel resulting from an SEM fit using OpenMx.
`conf_int`	Logical, whether to include confidence intervals in the output.
`conf_level`	The confidence level to use for the confidence intervals.
`...`	Additional arguments (currently not used).

Value

A tibble with one row per parameter and columns for parameter names, estimates, standard errors, and optionally confidence intervals.

See Also

OpenMx::OpenMx OpenMx::MxModel OpenMx::summary.MxModel

Examples

```
## Not run:
# Load Holzinger and Swineford (1939) dataset
data("HolzingerSwineford1939", package = "lavaan")
# Simple SEM model specification with OpenMx
manifestVars <- paste0("x", 1:9)
latVar <- c("visual", "textual", "speed")
model <- mxModel("Simple SEM",
  type = "RAM",
  manifestVars = manifestVars,
  latentVars = latVar,
  mxPath(from = "visual", to = c("x1", "x2", "x3")),
  mxPath(from = "textual", to = c("x4", "x5", "x6")),
  mxPath(from = "speed", to = c("x7", "x8", "x9")),
  mxPath(from = manifestVars, arrows = 2),
  mxPath(from = latVar, arrows = 2, free = FALSE, values = 1.0),
  mxPath(from = "one", to = manifestVars, arrows = 1, free = FALSE, values = 0),
  mxPath(from = "one", to = latVar, arrows = 1, free = FALSE, values = 0),
```

```
  mxData(HolzingerSwineford1939, type = "raw")
)
#
# Fit the model
fit0 <- mxRun(model)
tidy(fit0)

## End(Not run)
```

Index