Package 'wsMed'

May 8, 2026

Title Within-Subject Mediation Analysis Using Structural Equation Modeling

Version 1.0.2

Description Within-subject mediation analysis using structural equation modeling. Examine how changes in an outcome variable between two conditions are mediated through one or more variables. Supports within-subject mediation analysis using the 'lavaan' package by Rosseel (2012) <doi:10.18637/jss.v048.i02>, and extends Monte Carlo confidence interval estimation to missing data scenarios using the 'semmcci' package by Pesigan and Cheung (2023) <doi:10.3758/s13428-023-02114-4>.

License GPL (>= 3)

Depends R (>= 4.1.0)

Encoding UTF-8

RoxygenNote 7.3.3

Suggests rmarkdown, devtools, testthat

Config/testthat/edition 3

Imports knitr, lavaan, semmcci, mice, semboottools, MASS, rlang, dplyr, ggplot2, methods, stats

VignetteBuilder knitr

LazyData true

URL https://yangzhen1999.github.io/wsMed/

BugReports https://github.com/Yangzhen1999/wsMed/issues

NeedsCompilation no

Author Wendie Yang [aut, cre] (ORCID: <https://orcid.org/0009-0000-8388-6481>), Shu Fai Cheung [aut] (ORCID: <https://orcid.org/0000-0002-9871-9448>)

Maintainer Wendie Yang <1581075494q@gmail.com>

Repository CRAN

Date/Publication 2025-12-11 13:30:02 UTC

Contents

`GenerateModelCN` *Generate Chained Mediation Model*

Description

Dynamically generates a structural equation modeling (SEM) syntax for chained mediation analysis based on the prepared dataset. The function computes regression equations for mediators and the outcome variable, indirect effects along multi-step mediation paths, total effects, contrasts between indirect effects, and coefficients in different conditions.

Usage

```
GenerateModelCN(prepared_data, MP = character(0))
```

Arguments

`prepared_data`	A data frame returned by `PrepareData()`, containing the processed within-subject mediator and outcome variables. The data frame must include columns for difference scores (`Mdiff`) and average scores (`Mavg`) of mediators, as well as the outcome difference score (`Ydiff`).
`MP`	A character vector specifying which paths are moderated by variable(s) W. Valid entries include: - `"a2"`, `"a3"`, ...: moderation on the a paths (W → Mdiff), for mediators beyond M1. - `"b2"`, `"b3"`, ...: moderation on the b paths (Mdiff × W → Ydiff). - `"b_1_2"`, `"b_2_3"`, ...: moderation on cross-paths from one mediator to the next (e.g., M1 → M2). - `"d2"`, `"d3"`, ...: moderation on the d paths (Mavg × W → Ydiff). - `"d_1_2"`, `"d_2_3"`, ...: moderation on cross-paths from one Mavg to the next Mdiff. - `"cp"`: moderation on the direct effect from X to Y (i.e., W → Ydiff).

The function detects and inserts the correct interaction terms (e.g., \code{int_M2d

Details

This function is used to construct SEM models for chained mediation analysis. It automatically parses variable names from the prepared dataset and dynamically creates the necessary model syntax, including:

- **Outcome regression**: Defines the relationship between the difference scores of the outcome (`Ydiff`) and the mediators (`Mdiff`) as well as their average scores (`Mavg`).
- **Mediator regressions**: Defines the sequential regression models for each mediator's difference score, incorporating prior mediators as predictors.
- **Indirect effects**: Computes the indirect effects along all possible multi-step mediation paths using the product of path coefficients.
- **Total indirect effect**: Calculates the sum of all indirect effects from the chained mediation paths.
- **Total effect**: Combines the direct effect (`cp`) and the total indirect effect.
- **Contrasts of indirect effects**: Optionally calculates the pairwise contrasts between the indirect effects for different mediation paths.
- **Coefficients in different 'X' conditions**: Calculates path coefficients in different `X` conditions to observe the moderation effect of `X`.

This model is suitable for chained mediation designs where mediators influence each other in a sequential manner, forming multi-step mediation paths.

Value

A character string representing the SEM model syntax for the specified chained mediation analysis.

See Also

`PrepareData()`, `wsMed()`, `GenerateModelP()`

Examples

```
# Example prepared data
prepared_data <- data.frame(
  M1diff = rnorm(100),
  M2diff = rnorm(100),
  M3diff = rnorm(100),
  M1avg = rnorm(100),
  M2avg = rnorm(100),
  M3avg = rnorm(100),
  Ydiff = rnorm(100)
)

# Generate SEM model syntax
sem_model <- GenerateModelCN(prepared_data)
cat(sem_model)
```

GenerateModelCP *Generate Combined Parallel and Chained Mediation Model*

Description

Dynamically generates a structural equation modeling (SEM) syntax for combined parallel and chained mediation analysis based on the prepared dataset. The function computes regression equations for mediators and the outcome variable, indirect effects for both parallel and chained mediation paths, total effects, contrasts between indirect effects, and coefficients in different X conditions.

Usage

```
GenerateModelCP(prepared_data, MP = character(0))
```

Arguments

prepared_data	A data frame returned by PrepareData(), containing the processed within-subject mediator and outcome variables. The data frame must include columns for difference scores (Mdiff) and average scores (Mavg) of mediators, as well as the outcome difference score (Ydiff).
MP	A character vector specifying which paths are moderated by variable(s) W. Acceptable values include: - "a2", "a3", ...: moderation on the a paths of parallel mediators (W → Mdiff). - "b2", "b3", ...: moderation on the b paths of parallel mediators (Mdiff × W → Ydiff). - "b_1_2", "b_1_3", ...: moderation on the paths from the chain mediator to parallel mediators. - "d_1_2", "d_1_3", ...: moderation on the paths from the chain mediator's Mavg to parallel mediators. - "cp": moderation on the direct effect from X to Y (i.e., W → Ydiff).

Each entry triggers inclusion of W's main effect or interaction terms (e.g., \code{i

Details

This function is used to construct SEM models that combine parallel and chained mediation analysis. It automatically parses variable names from the prepared dataset and dynamically creates the necessary model syntax, including:

- **Outcome regression**: Defines the relationship between the difference scores of the outcome (Ydiff), the chained mediator (M1diff), and the parallel mediators (M2diff, M3diff, etc.).
- **Mediator regressions**: Defines the sequential regression models for the chained mediator and each parallel mediator.
- **Indirect effects**: Computes the indirect effects for both chained and parallel mediation paths, including multi-step indirect effects involving both chained and parallel mediators.
- **Total indirect effect**: Calculates the sum of all indirect effects from chained and parallel mediation paths.
- **Total effect**: Combines the direct effect (cp) and the total indirect effect.
- **Contrasts of indirect effects**: Optionally calculates the pairwise contrasts between the indirect effects for different mediation paths.

- **Coefficients in different 'X' conditions**: Calculates path coefficients in different `X` conditions to observe the moderation effect of `X`.

This model is suitable for designs where mediators include both a sequential chain (chained mediation) and independent parallel mediators.

Value

A character string representing the SEM model syntax for the specified combined parallel and chained mediation analysis.

See Also

`PrepareData()`, `wsMed()`, `GenerateModelP()`, `GenerateModelCN()`

Examples

```
# Example prepared data
prepared_data <- data.frame(
  M1diff = rnorm(100),
  M2diff = rnorm(100),
  M3diff = rnorm(100),
  M1avg = rnorm(100),
  M2avg = rnorm(100),
  M3avg = rnorm(100),
  Ydiff = rnorm(100)
)

# Generate SEM model syntax
sem_model <- GenerateModelCP(prepared_data)
cat(sem_model)
```

`GenerateModelP` *Generate Parallel Mediation Model*

Description

Dynamically generates a structural equation modeling (SEM) syntax for parallel mediation analysis based on the prepared dataset. The function computes regression equations for mediators and the outcome variable, indirect effects, total effects, contrasts between indirect effect, and .

Usage

```
GenerateModelP(prepared_data, MP = character(0))
```

Arguments

prepared_data	A data frame returned by PrepareData(), containing the processed within-subject mediator and outcome variables. The data frame must include columns for difference scores (Mdiff) and average scores (Mavg) of mediators, as well as the outcome difference score (Ydiff).
MP	A character vector specifying which paths are moderated by variable(s) W. Valid values include: - "a1", "a2", ...: moderation on the a paths (W → Mdiff). - "b1", "b2", ...: moderation on the b paths (Mdiff × W → Ydiff). - "d1", "d2", ...: moderation on the d paths (Mavg × W → Ydiff). - "cp": moderation on the direct effect from X to Y (i.e., W → Ydiff).

```
This argument controls which interaction terms (e.g., \code{int_Mdiff_W}, \code{in
  added to the corresponding regression equations.
```

Details

This function is used to construct SEM models for parallel mediation analysis. It automatically parses variable names from the prepared dataset and dynamically creates the necessary model syntax, including:

- **Outcome regression**: Defines the relationship between the difference scores of the outcome (Ydiff) and the mediators (Mdiff) as well as their average scores (Mavg).
- **Mediator regressions**: Defines the intercept models for each mediator's difference score.
- **Indirect effects**: Computes the indirect effects for each mediator using the product of path coefficients (e.g., a * b).
- **Total indirect effect**: Calculates the sum of all indirect effects.
- **Total effect**: Combines the direct effect (cp) and the total indirect effect.
- **Contrasts of indirect effects**: Optionally calculates the pairwise contrasts between the indirect effects when multiple mediators are present.
- **coefficients in different 'X' conditions**: Calculates path coefficients in different X conditions to observe the moderation effect of 'X'.

This model is suitable for parallel mediation designs where multiple mediators act independently.

Value

A character string representing the SEM model syntax for the specified parallel mediation analysis.

See Also

PrepareData(), wsMed(), GenerateModelCN()

Examples

```
# Example prepared data
prepared_data <- data.frame(
  M1diff = rnorm(100),
  M2diff = rnorm(100),
```

```
  M1avg = rnorm(100),
  M2avg = rnorm(100),
  Ydiff = rnorm(100)
)

# Generate SEM model syntax
sem_model <- GenerateModelP(prepared_data)
cat(sem_model)
```

GenerateModelPC *Generate Parallel and Chained Mediation Model*

Description

Dynamically generates a structural equation modeling (SEM) syntax for mediation analysis that integrates both parallel and chained mediators. Unlike the Combined Parallel and Chained mediation model (`GenerateModelCP`), this function assumes that the chained mediator receives inputs from the parallel mediators and directly influences the outcome variable, emphasizing a downstream role for the chained mediator.

Usage

```
GenerateModelPC(prepared_data, MP = character(0))
```

Arguments

`prepared_data`	A data frame returned by `PrepareData()`, containing the processed within-subject mediator and outcome variables. The data frame must include columns for difference scores (`Mdiff`) and average scores (`Mavg`) of mediators, as well as the outcome difference score (`Ydiff`).
`MP`	A character vector specifying which paths are moderated by variable(s) W. Acceptable values include: - `"a2"`, `"a3"`, ...: moderation on the a paths (W → Mdiff). - `"b2"`, `"b3"`, ...: moderation on the b paths (Mdiff × W → Ydiff). - `"b_2_1"`, `"b_3_1"`, ...: moderation on the cross-paths from parallel mediators to the chain mediator (e.g., M2 → M1). - `"d_2_1"`, `"d_3_1"`, ...: moderation on the paths from parallel mediators' centered means to M1. - `"cp"`: moderation on the direct effect of X on Y (i.e., W → Ydiff).

```
This argument controls which interaction terms (e.g., \code{int_Mdiff_W}, \code{int
in the regression equations. Variable names are automatically matched using the nam
  \code{"int_<predictor>_W<index>"}.
```

Details

This function is designed to build SEM models that integrate parallel and chained mediation structures. It automatically identifies variable names from the prepared dataset and generates the necessary model syntax, including:

- **Outcome regression**: Defines the relationship between the difference scores of the outcome (`Ydiff`), the chained mediator (`M1diff`), and the parallel mediators (`M2diff`, `M3diff`, etc.).
- **Mediator regressions**: Constructs separate regression models for the parallel mediators and the chained mediator. The chained mediator incorporates predictors from all parallel mediators.
- **Indirect effects**: Computes indirect effects for:
  - Parallel mediators (`M2diff`, `M3diff`, etc.) directly influencing the outcome.
  - The chained mediator (`M1diff`) directly influencing the outcome.
  - Combined paths where parallel mediators influence the chained mediator, which in turn influences the outcome.
- **Total indirect effect**: Summarizes all indirect effects from parallel and chained mediation paths.
- **Total effect**: Combines the direct effect (`cp`) and the total indirect effect.
- **Contrasts of indirect effects**: Optionally computes pairwise contrasts between indirect effects for different mediation paths.
- **Coefficients in different 'X' conditions**: Computes path coefficients under different `X` conditions to analyze moderation effects.

This model is suitable for designs where mediators include both independent parallel paths and sequential chained paths, providing a comprehensive mediation analysis framework.

Value

A character string representing the SEM model syntax for the specified parallel and chained mediation analysis.

See Also

`PrepareData()`, `wsMed()`, `GenerateModelP()`, `GenerateModelCN()`

Examples

```
# Example prepared data
prepared_data <- data.frame(
  M1diff = rnorm(100),
  M2diff = rnorm(100),
  M3diff = rnorm(100),
  M1avg = rnorm(100),
  M2avg = rnorm(100),
  M3avg = rnorm(100),
  Ydiff = rnorm(100)
)

# Generate SEM model syntax
sem_model <- GenerateModelPC(prepared_data)
cat(sem_model)
```

ImputeData *Impute Missing Data Using Multiple Imputation*

Description

The ImputeData function performs multiple imputation on a data frame with missing values using the mice package. It handles missing data by creating multiple imputed datasets based on a specified imputation method and returns a list of completed data frames.

Usage

```
ImputeData(
  data_missing,
  m = 5,
  method = "pmm",
  seed = 123,
  predictorMatrix = NULL
)
```

Arguments

data_missing	A data frame containing missing values to be imputed. The function replaces values coded as -999 with NA before imputation.
m	An integer specifying the number of imputed datasets to generate.
method	A character string specifying the imputation method. Default is "pmm" (predictive mean matching).
seed	An integer for setting the random seed to ensure reproducibility. Default is 123.
predictorMatrix	An optional matrix specifying the predictor structure for the imputation model. Default is NULL, meaning that the function will use the default predictor matrix created by mice.

Details

This function replaces specified missing value placeholders (e.g., -999) with NA, and then applies the multiple imputation by chained equations (MICE) procedure to generate multiple imputed datasets. It supports flexible imputation methods and allows for specifying a custom predictor matrix.

Value

A list of m imputed data frames.

Author(s)

Wendie Yang, Shufai Cheung

Examples

```
# Example data with missing values
data <- data.frame(
  M1 = c(rnorm(99), rep(NA, 1)),
  M2 = c(rnorm(99), rep(NA, 1)),
  Y1 = rnorm(100),
  Y2 = rnorm(100)
)
# Perform multiple imputation
imputed_data_list <- ImputeData(data, m = 5)
# Display the first imputed dataset
head(imputed_data_list[[1]])
```

`MCMI2` *Monte Carlo Confidence Intervals for Multiple Imputation SEM Models*

Description

Computes Monte Carlo confidence intervals (MCCI) for structural equation models (SEM) fitted to multiple imputed datasets. This function integrates SEM fitting across imputed datasets, pools the results, and generates confidence intervals through Monte Carlo sampling.

Usage

```
MCMI2(
  sem_model,
  imputations,
  R = 20000L,
  alpha = c(0.001, 0.01, 0.05),
  decomposition = "eigen",
  pd = TRUE,
  tol = 1e-06,
  seed = NULL,
  estimator = "ML",
  se = "standard",
  missing = "listwise"
)
```

Arguments

`sem_model`	A character string specifying the SEM model syntax.
`imputations`	A list of data frames, where each data frame represents an imputed dataset.
`R`	An integer specifying the number of Monte Carlo samples. Default is `20000L`.
`alpha`	A numeric vector specifying significance levels for the confidence intervals. Default is `c(0.001, 0.01, 0.05)`.

decomposition	A character string specifying the decomposition method for the covariance matrix. Default is "eigen". Options include "chol", "eigen", or "svd".
pd	A logical value indicating whether to ensure positive definiteness of the covariance matrix. Default is TRUE.
tol	A numeric value specifying the tolerance for positive definiteness checks. Default is 1e-06.
seed	An optional integer specifying the random seed for reproducibility. Default is NULL.
estimator	A character string specifying the estimator for SEM fitting. Default is "ML" (Maximum Likelihood).
se	A character string specifying the type of standard errors to compute. Default is "standard".
missing	A character string specifying the method for handling missing data in SEM fitting. Default is "listwise".

Details

This function is designed for SEM models that require multiple imputation to handle missing data. It performs the following steps:

- **SEM Fitting**: Fits the specified SEM model to each imputed dataset using lavaan::sem().
- **Pooling Results**: Combines parameter estimates and covariance matrices across imputations using Rubin's rules.
- **Monte Carlo Sampling**: Generates Monte Carlo samples based on the pooled estimates and covariance matrices, and calculates confidence intervals for model parameters.

This function supports custom estimators, handling of missing data, and precision adjustments for Monte Carlo sampling. It is particularly useful for mediation analysis or complex SEM models where missing data are addressed using multiple imputation.

Value

An object of class semmcci containing:

- call: The matched function call.
- args: A list of input arguments.
- thetahat: The pooled parameter estimates.
- thetahatstar: Monte Carlo samples for parameter estimates.
- fun: The name of the function ("MCMI2").

See Also

lavaan::sem(), semmcci::MC(), semmcci::MCStd()

Examples

```
# Example SEM model
sem_model <- "
  Ydiff ~ b1 * M1diff + cp * 1
  M1diff ~ a1 * 1
  indirect := a1 * b1
  total := cp + indirect
"

# Example imputed datasets
imputations <- list(
  data.frame(M1diff = rnorm(100), Ydiff = rnorm(100)),
  data.frame(M1diff = rnorm(100), Ydiff = rnorm(100))
)

# Compute Monte Carlo confidence intervals
result <- MCMI2(
  sem_model = sem_model,
  imputations = imputations,
  R = 1000,
  alpha = c(0.05, 0.01),
  seed = 123
)
```

`plot_moderation_curve` *Plot moderation curves with Johnson-Neyman highlights*

Description

`plot_moderation_curve()` visualises how an indirect effect (`theta_curve`) or a path coefficient (`path_curve`) varies along a continuous moderator *W*.

The routine

- extracts the requested record (`path_name`) from `result$moderation`, preferring `theta_curve` when it is available in both curves;
- draws the conditional effect (`Estimate`) against the raw moderator grid (`W_raw`);
- overlays the Monte-Carlo confidence band (`CI.LL`, `CI.UL`) and finds every Johnson–Neyman segment whose 95 % CI excludes zero (`CI.LL * CI.UL > 0`);
- shades these significant regions and annotates each with its start / end percentiles (for example, `"sig 12.5%-38.3%"`).

Visual elements

- **Red ribbon** – overall 95 % confidence band (`ns_fill`);
- **Green ribbon** – significant Johnson–Neyman intervals (`sig_fill`);
- **Solid line** – point estimate;
- **Dashed h-line** – zero reference;
- **Dashed v-lines** – J–N bounds.

Usage

```
plot_moderation_curve(
  result,
  path_name,
  title = NULL,
  x_label = "Moderator (W)",
  y_label = "Estimate",
  ns_fill = "#FEE0D2",
  sig_fill = "#C7E9C0",
  alpha_ci = 0.35,
  alpha_sig = 0.35,
  base_size = 14
)
```

Arguments

result	A wsMed() result that contains a $moderation element.
path_name	Exact name of the path to plot (e.g. "indirect_1_2" or "b_1_2"). When the name exists in both curves, theta_curve is used.
title	Optional plot title (default sprintf("Effect Curve: (%s)", path_name)).
x_label, y_label	Axis labels. Defaults are "Moderator (W)" and "Estimate".
ns_fill, sig_fill	Fill colours for the confidence band and the significant regions.
alpha_ci, alpha_sig	Alpha values for the two ribbons.
base_size	Base font size passed to ggplot2::theme_minimal().

Value

A ggplot object (add layers or save with ggsave()).

PrepareData *Prepare Data for Two-Condition Within-Subject Mediation (WsMed)*

Description

PrepareData() transforms raw pre/post data into the set of variables required by the **WsMed** workflow. It handles *mediators*, *outcome*, *within-subject controls*, *between-subject controls*, *moderators*, and all necessary **interaction terms**, while automatically centering / dummy-coding variables as needed.

Usage

```
PrepareData(
  data,
  M_C1,
  M_C2,
  Y_C1,
  Y_C2,
  C_C1 = NULL,
  C_C2 = NULL,
  C = NULL,
  C_type = NULL,
  W = NULL,
  W_type = NULL,
  center_W = TRUE,
  keep_W_raw = TRUE,
  keep_C_raw = TRUE
)
```

Arguments

data	A data frame with the raw pre/post measures.
M_C1, M_C2	Character vectors: mediator names at occasion 1 and 2 (equal length).
Y_C1, Y_C2	Character scalars: outcome names at occasion 1 and 2.
C_C1, C_C2	Optional character vectors: within-subject control names.
C	Optional character vector: between-subject control names.
C_type	Optional vector of the same length as C. Each element is one of "continuous", "categorical", or "auto" (default). Ignored when C = NULL.
W	Optional character vector: moderator names (one or more).
W_type	Optional vector of the same length as W. Same coding as C_type. Ignored when W = NULL.
center_W	Logical. Whether to center the moderator variable W.
keep_W_raw, keep_C_raw	Logical. If TRUE, keep the original W / C columns in the returned data.

Details

The function performs the following steps:

1. Outcome difference: Ydiff = Y_C2 - Y_C1.
2. Mediator variables for each pair (M_C1[i], M_C2[i]):
   - Mi_diff = M_C2 - M_C1
   - Mi_avg is the mean-centered average of the two occasions.
3. Between-subject controls C:
   - Continuous variables are grand-mean centered (Cb1, Cb2, ...).

- Categorical variables (binary or multi-level) are expanded into `k - 1` dummy variables (`Cb1_1`, `Cb2_1`, `Cb2_2`, ...), using the first level as the reference.

4. Within-subject controls `Cw`: difference and centered-average versions (`Cw1diff`, `Cw1avg`, ...).
5. Moderators `W` (one or more):
   - Continuous variables are grand-mean centered (`W1`, `W2`, ...).
   - Categorical variables are dummy-coded in the same way as `C`.
6. Interaction terms between each moderator column and each mediator column:
   - `int_<Mi_diff>_<Wj>`, `int_<Mi_avg>_<Wj>`.
7. Two attributes are added to the returned data:
   - `"W_info"`: raw names, dummy names, level mapping
   - `"C_info"`: same structure for between-subject controls.

Row counts are preserved even if input factors contain NA values (model.matrix is called with `na.action = na.pass`).

Value

A data frame containing at minimum:

- `Ydiff`
- `Mi_diff`, `Mi_avg` for each mediator
- centered or dummy-coded `Cb*`, `Cw*diff`, `Cw*avg`
- centered or dummy-coded `W*` and all `int_*` interaction terms

plus the attributes `"W_info"` and `"C_info"` described above.

See Also

PrepareMissingData, GenerateModelP, wsMed

Examples

```
set.seed(1)
raw <- data.frame(
  A1 = rnorm(50), A2 = rnorm(50),   # mediator 1
  B1 = rnorm(50), B2 = rnorm(50),   # mediator 2
  C1 = rnorm(50), C2 = rnorm(50),   # outcome
  D1 = rnorm(50), D2 = rnorm(50),   # within-subject control
  W_bin  = sample(0:1, 50, TRUE),   # between-subject binary C
  W_fac3 = factor(sample(c("Low","Med","High"), 50, TRUE)) # moderator W
)

prep <- PrepareData(
  data  = raw,
  M_C1  = c("A1","B1"), M_C2 = c("A2","B2"),
  Y_C1  = "C1",         Y_C2 = "C2",
  C_C1  = "D1",         C_C2 = "D2",
  C     = "W_bin",      C_type = "categorical",
```

```
  W     = "W_fac3",     W_type = "categorical"
)
head(prep)
```

`PrepareMissingData` *Prepare Data with Missing Values for Mediation Analysis*

Description

Handles missing values in the dataset through multiple imputation and prepares the imputed datasets for within-subject mediation analysis. The function imputes missing data, processes each imputed dataset, and provides diagnostics for the imputation process.

Usage

```
PrepareMissingData(
  data_missing,
  m = 5,
  method_num = "pmm",
  seed = 123,
  M_C1,
  M_C2,
  Y_C1,
  Y_C2,
  C_C1 = NULL,
  C_C2 = NULL,
  C = NULL,
  C_type = NULL,
  W = NULL,
  W_type = NULL,
  center_W = TRUE,
  keep_W_raw = TRUE,
  keep_C_raw = TRUE
)
```

Arguments

`data_missing`	A data frame containing the raw dataset with missing values.
`m`	An integer specifying the number of imputations to perform. Default is `5`.
`method_num`	Character; imputation method for numeric variables (for example, `"pmm"`, `"norm"`). Default is `"pmm"`.
`seed`	An integer specifying the random seed for reproducibility. Default is `123`.
`M_C1`	A character vector of column names representing mediators at condition 1.
`M_C2`	A character vector of column names representing mediators at condition 2. Must match the length of `M_C1`.

Y_C1	A character string representing the column name of the outcome variable at condition 1.
Y_C2	A character string representing the column name of the outcome variable at condition 2.
C_C1	Character vector of within-subject control variable names (condition 1).
C_C2	Character vector of within-subject control variable names (condition 2).
C	Character vector of between-subject control variable names.
C_type	Optional vector of the same length as C. Each element is "continuous", "categorical", or "auto" (default). Ignored when C = NULL.
W	Optional character vector: moderator names (at most J).
W_type	Optional vector of the same length as W. Same coding as C_type. Ignored when W = NULL.
center_W	Logical. Whether to center the moderator variable W.
keep_W_raw, keep_C_raw	Logical; keep the original W / C columns in the returned data?

Details

This function is designed to preprocess datasets with missing values for mediation analysis. It performs the following steps:

- Multiple imputation: Uses specified imputation methods (for example, predictive mean matching) to generate m imputed datasets.
- Data preparation: Applies PrepareData to each of the m imputed datasets to calculate difference scores and centered averages for mediators and the outcome variable.
- Imputation diagnostics: Provides summary diagnostics for the imputation process, including information about missing data patterns and convergence.

This function integrates imputation and data preparation, ensuring that the resulting datasets are ready for subsequent mediation analysis.

Value

A list containing:

processed_data_list A list of m data frames, each representing an imputed and processed dataset ready for within-subject mediation analysis.

imputation_summary A summary of the imputation process, including diagnostics and convergence information.

See Also

PrepareData, ImputeData, wsMed

Examples

```
# Example dataset with missing values
data("example_data", package = "wsMed")
set.seed(123)
example_dataN <- mice::ampute(
  data = example_data,
  prop = 0.1
)$amp

# Prepare the dataset with multiple imputations
prepared_missing_data <- PrepareMissingData(
  data_missing = example_dataN,
  m = 5,
  M_C1 = c("A2", "B2"),
  M_C2 = c("A1", "B1"),
  Y_C1 = "C2",
  Y_C2 = "C1"
)

# Access processed datasets
processed_data_list <- prepared_missing_data$processed_data_list
imputation_summary  <- prepared_missing_data$imputation_summary
```

---

print.wsMed *Print Method for wsMed Objects*

Description

Provides a comprehensive summary of results from a wsMed object, including:

- Input and computed variables with sample size.
- Model fit indices, regression paths, and variance estimates.
- Total, direct, and indirect effects with pairwise contrasts.
- Moderation effects and Monte Carlo confidence intervals for raw and standardized estimates (if applicable).
- Diagnostic notes for bootstrapping, imputation, and analysis parameters.

The output is formatted for clarity, ensuring an intuitive presentation of mediation analysis results, including dynamic confidence intervals, moderation keys, and C1-C2 coefficients.

Usage

```
## S3 method for class 'wsMed'
print(x, digits = 3, ...)
```

Arguments

x	A wsMed object containing the results of within-subject mediation analysis.
digits	Numeric. Number of digits to display in the results.
...	Additional arguments (not used currently).

Details

This function is specifically designed to display results from the within-subject mediation analysis conducted using the wsMed function. Key features include:

- **Variables**:
  - Shows input variables (M_C1, M_C2, Y_C1, Y_C2) and computed variables like Ydiff, Mdiff, and Mavg.
  - Reports the sample size used in the analysis.
- **Model Fit Indices**:
  - Displays SEM fit indices (e.g., Chi-square, CFI, TLI, RMSEA, SRMR) to assess model quality.
- **Regression Paths and Variance Estimates**:
  - Summarizes path coefficients, intercepts, variances, and confidence intervals.
- **Effects**:
  - Reports total, direct, and indirect effects with their significance.
  - Highlights pairwise contrasts between indirect effects for mediation paths.
- **Moderation Effects**:
  - Provides moderation results for identified variables with corresponding coefficients and paths.
- **Monte Carlo Confidence Intervals**:
  - Includes results for raw and standardized estimates obtained using methods such as MI or FIML.
- **Diagnostics**:
  - Summarizes analysis parameters like bootstrapping, imputation settings, Monte Carlo iterations, and random seeds.

Value

Invisibly returns the input wsMed object for further use.

See Also

wsMed, sem, standardizedSolution_boot_ci

Examples

```
# Perform within-subject mediation analysis
data("example_data", package = "wsMed")
result1 <- wsMed(
  data = example_data,
  M_C1 = c("A1", "B1"),
  M_C2 = c("A2", "B2"),
  Y_C1 = "C1",
  Y_C2 = "C2",
  form = "P",
  Na = "FIML",
  standardized = FALSE,
  alpha = 0.05
)

# Print the results
print(result1)
```

---

printGM *Print Formatted SEM Model Syntax*

Description

Formats and prints the SEM model syntax generated by `GenerateModelCN`, `GenerateModelCP`, `GenerateModelP`, and `GenerateModelPC`. It organizes the equations into labeled sections for better readability.

Usage

```
printGM(x, ...)
```

Arguments

`x`	A list or character string containing SEM model syntax. If `x` is a `wsMed()` result, it will extract and print the model syntax. If `x` is a `GenerateModel*()` result, it will format and print the model.
`...`	Additional arguments (not used).

Value

Invisibly returns the formatted SEM model syntax.

Examples

```
data(example_data)
head(example_data)
prepared_data <- PrepareData(
  data = example_data,
  M_C1 = c("A1", "B1"),
  M_C2 = c("A2", "B2"),
  Y_C1 = "C1",
  Y_C2 = "C2"
)
sem_model <- GenerateModelPC(prepared_data)
printGM(sem_model)
```

---

wsMed *Within-Subject Mediation Analysis (Two-Condition)*

Description

`wsMed()` fits a structural equation model (SEM) for two-condition within-subject mediation. It can handle missing data (DE, FIML, MI) and computes both unstandardized and standardized effects with bootstrap or Monte Carlo confidence intervals.

Usage

```
wsMed(
  data,
  M_C1,
  M_C2,
  Y_C1,
  Y_C2,
  C_C1 = NULL,
  C_C2 = NULL,
  C = NULL,
  C_type = NULL,
  W = NULL,
  W_type = NULL,
  MP = NULL,
  form = c("P", "CN", "CP", "PC"),
  Na = c("DE", "FIML", "MI"),
  alpha = 0.05,
  mi_args = list(),
  R = 20000L,
  bootstrap = 2000,
  boot_ci_type = "perc",
  iseed = 123,
  fixed.x = FALSE,
  ci_method = c("mc", "bootstrap", "both"),
```

```
  MCmethod = NULL,
  seed = 123,
  standardized = FALSE,
  verbose = FALSE
)
```

Arguments

data	A data.frame containing the raw scores.
M_C1, M_C2	Character vectors of mediator names under condition 1 and 2.
Y_C1, Y_C2	Character scalars for the outcome under each condition.
C_C1, C_C2	Character vectors of within-subject covariates (per condition).
C	Character vector of between-subject covariates.
C_type	Character; type of C: "continuous" or "categorical".
W	Character vector of moderators. Default NULL.
W_type	Character; "continuous" or "categorical".
MP	Character vector identifying which regression paths are moderated (for example, "a1", "b_1_2", "cp").
form	Model type: "P", "CN", "CP", or "PC".
Na	Missing-data method: "DE", "FIML", or "MI".
alpha	Numeric vector in (0, 1); two-sided significance levels.
mi_args	List of MI-specific controls: m Number of imputations. Default 5. method_num Imputation method for mice(). decomposition Covariance-decomposition method ("eigen", "chol", "svd"). pd Logical; positive-definiteness check. tol Tolerance for the positive-definiteness check.
R	Integer; number of Monte Carlo draws. Default 20000L.
bootstrap	Integer; number of bootstrap replicates (DE and FIML only).
boot_ci_type	Character; bootstrap CI type: "perc", "bc", or "bca.simple".
iseed, seed	Integer seeds for bootstrap and Monte Carlo, respectively.
fixed.x	Logical; passed to **lavaan**.
ci_method	CI engine: "bootstrap" or "mc". If NULL (default) the choice is "bootstrap" for Na = "DE" and "mc" otherwise.
MCmethod	If Na = "FIML" and ci_method = "mc", choose "mc" (default) or "bootSD".
standardized	Logical; if TRUE, return standardized effects. Default FALSE.
verbose	Logical; print progress messages.

Details

Model structures:

- "P": parallel mediation
- "CN": chained (serial) mediation
- "CP": chained then parallel
- "PC": parallel then chained

Missing-data strategies:

- "DE": list-wise deletion
- "FIML": full-information maximum likelihood
- "MI": multiple imputation via **mice**

Confidence-interval engines:

- Bootstrap: percentile, BC, or BCa (DE and FIML only)
- Monte Carlo: draws via **semmcci** (all Na options)

For Na = "FIML", you may choose MCmethod = "mc" (default) or "bootSD" to add a finite-sample SD correction.

Workflow: (1) preprocess -> (2) generate SEM syntax -> (3) fit -> (4) compute confidence intervals -> (5) optional: standardize estimates.

Value

An object of class "wsMed" with elements:

data Preprocessed data frame.

sem_model Generated **lavaan** syntax.

mc List with Monte Carlo draws, bootstrap tables (if any), and the fitted model.

moderation Conditional or moderated effect tables.

form,Na,alpha Analysis settings.

input_vars Names of all user-supplied variables.

Examples

```
data("example_data", package = "wsMed")
set.seed(123)
result <- wsMed(
  data = example_data,
  M_C1 = c("A2", "B2"),
  M_C2 = c("A1", "B1"),
  Y_C1 = "C1", Y_C2 = "C2",
  form = "P", Na = "DE"
)
print(result)
```

Index