Package: wsMed (via r-universe)

October 28, 2026

Title Within-Subject Mediation Analysis Using Structural Equation Modeling

Version 1.2.0

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 7.3.2

Suggests rmarkdown, devtools, testthat, smcfcs (>= 1.8.0)

Config/testthat/edition 3

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

VignetteBuilder knitr

LazyData true

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

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

NeedsCompilation no

Maintainer Wendie Yang <1581075494q@gmail.com>

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>)

Repository https://cran.r-universe.dev

Date/Publication 2026-10-28 09:52:20 UTC

RemoteUrl https://github.com/cran/wsMed

RemoteRef HEAD

RemoteSha 9645d50d8736fa000fcbab21115b22bf7174d832

Contents

example_data . . . 2
GenerateModelCN . . . 3
GenerateModelCP . . . 5
GenerateModelCustom . . . 7
GenerateModelP . . . 8
GenerateModelPC . . . 10
ImputeData . . . 12
MCMI2 . . . 13
plot_conditional_effects . . . 15
plot_contrasts . . . 17
plot_effects . . . 18
plot_moderation_curve . . . 19
PrepareData . . . 21
PrepareMissingData . . . 23
printGM . . . 25
standardize_moderation . . . 26
wsMed . . . 27
wsmed_categorical . . . 31
wsmed_contrasts . . . 32
wsmed_effects . . . 33
wsmed_examples . . . 34
wsmed_fit . . . 36
wsmed_infer . . . 37
wsmed_inspect . . . 38
wsmed_methods . . . 39
wsmed_model . . . 42
wsmed_plots . . . 43
wsmed_reproducibility . . . 47

Index **48**

`example_data` *Simulated data for mediation tutorials*

Description

Artificial data used to demonstrate model syntax, estimation, and categorical predictors. The labels do not identify substantive psychological constructs. This is distinct from the empirical manuscript dataset `wsmed_examples()`.

Usage

```
example_data
```

Format

A tibble (data frame) with 100 rows and 14 variables:

A1, A2, A3 Numeric measurements for variable A at three conditions.

B1, B2, B3 Numeric measurements for variable B at three conditions.

C1, C2, C3 Numeric measurements for variable C at three conditions.

D1, D2, D3 Numeric measurements for variable D at three conditions.

Group An unordered factor with levels high, med, and low, in that order.

W_Group An unordered factor with levels high, med, and low, in that order.

Details

All columns are complete. A wsMed analysis selects two condition columns per outcome or mediator. The factor labels are nominal categories; they do not request an ordinal regression model. Set their level order explicitly before fitting to select a different reference category.

See Also

`wsmed_examples()`, wsmed_categorical

Examples

```
data(example_data)
dim(example_data)
levels(example_data$Group)
```

GenerateModelCN *Generate a serial 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.

Product moments

When a moderated path starts at an endogenous mediator, auxiliary product moment equations and covariances with its upstream disturbances preserve the joint predictor moment structure under reference recoding. These equations are not additional mediation paths. Genuine exogenous moments follow the fitting function's fixed.x setting; endogenous product moments are estimated.

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 a serial-parallel 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.

Product moments

When a moderated path starts at an endogenous mediator, auxiliary product moment equations and covariances with its upstream disturbances preserve the joint predictor moment structure under reference recoding. These equations are not additional mediation paths. Genuine exogenous moments follow the fitting function's `fixed.x` setting; endogenous product moments are estimated.

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)
```

`GenerateModelCustom` *Generate a user-defined mediation model*

Description

Generates lavaan model syntax for a user-defined within-subject mediation model. Users specify directed paths among mediators and from mediators to the outcome. The function automatically adds the corresponding difference and level components and identifies all indirect effects.

Usage

```
GenerateModelCustom(prepared_data, paths, MP = character(0))
```

Arguments

`prepared_data`	A data frame returned by `PrepareData()`. It must contain `M1diff`, `M1avg`, ..., and `Ydiff`.
`paths`	A character vector defining directed paths among mediators and from mediators to the outcome. For example: `c("M1 -> M2", "M2 -> Y")`.
`MP`	A character vector identifying moderated paths. Supported labels follow the existing wsMed convention: `a1`, `b1`, `d1`, `b_1_2`, `d_1_2`, and `cp`. Main effects of W are always labeled and used for conditional intercepts, including when a/cp is not explicitly listed in MP.

Value

A character string containing lavaan model syntax.

Product moments

When a moderated path starts at an endogenous mediator, auxiliary product moment equations and covariances with its upstream disturbances preserve the joint predictor moment structure under reference recoding. These equations are not additional mediation paths. Genuine exogenous moments follow the fitting function's `fixed.x` setting; endogenous product moments are estimated.

Examples

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

model <- GenerateModelCustom(
 prepared_data = prepared_data,
 paths = c(
   "M1 -> M3",
   "M1 -> Y",
   "M3 -> Y",
   "M2 -> Y"
 )
)

cat(model)
```

GenerateModelP *Generate a 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{int
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.

Product moments

When a moderated path starts at an endogenous mediator, auxiliary product moment equations and covariances with its upstream disturbances preserve the joint predictor moment structure under reference recoding. These equations are not additional mediation paths. Genuine exogenous moments follow the fitting function's `fixed.x` setting; endogenous product moments are estimated.

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 a parallel-serial 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.

Product moments

When a moderated path starts at an endogenous mediator, auxiliary product moment equations and covariances with its upstream disturbances preserve the joint predictor moment structure under reference recoding. These equations are not additional mediation paths. Genuine exogenous moments follow the fitting function's `fixed.x` setting; endogenous product moments are estimated.

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",
  fixed.x = FALSE
)
```

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".
fixed.x	Whether exogenous covariates are treated as fixed in every imputation. Defaults to FALSE, consistently with wsMed().

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.

With `decomposition = "eigen"`, Monte Carlo sampling uses the principal symmetric covariance square root. Equivalent eigenvector signs and repeated-eigenvalue bases therefore produce the same draws up to numerical precision. Fixed-seed draws differ from wsMed 1.1.0; the target distribution and pooled point estimates are unchanged. Reproducing the complete analysis also requires matching data, imputation settings, and software dependencies.

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_conditional_effects

Plot conditional effects at moderator levels

Description

Draw point estimates and confidence intervals by categorical group or by the stored low, mean and high levels of a continuous moderator. Each effect is shown in its own panel. Categories are not joined by interpolating lines.

Usage

```
plot_conditional_effects(
  result,
  paths = NULL,
  type = c("indirect", "paths", "overall"),
  levels = NULL,
  standardized = FALSE,
  engine = c("mc", "boot"),
  labels = NULL,
```

```
  title = NULL,
  x_label = NULL,
  y_label = NULL,
  base_size = 12,
  ncol = NULL
)
```

Arguments

result	A result returned by wsMed().
paths	Optional character vector of effect identifiers, in display order. NULL selects all available effects. Both indirect_1 and indirect_effect_1 are accepted when they identify an available effect.
type	"indirect" for specific indirect effects, "paths" for available conditional path coefficients, or "overall" for total indirect and total effects.
levels	Optional group/level labels to select, in display order. NULL uses their stored order. For continuous W the labels are "-1 SD", "0 SD", "+1 SD"; the axis also displays stored raw W probe values.
standardized	Use standardized results, requested when fitting with standardized = TRUE. No scaling is recomputed from interval endpoints.
engine	For ci_method = "both", select "mc" or "boot". For a single engine its stored results are used.
labels	Optional named character vector mapping effect identifiers to display labels. Partial mappings are allowed.
title	Optional plot title.
x_label, y_label	Optional axis labels.
base_size	Base font size.
ncol	Optional number of facet columns.

Details

Intervals are the stored percentile conditional intervals, including jointly transformed scales for standardized results. They are individual intervals, not simultaneous confidence intervals.

Value

A ggplot object with plotted data and a wsmed_plot attribute.

See Also

plot_effects(), plot_contrasts()

plot_contrasts *Plot differences between mediation effects*

Description

Draw contrast estimates and their confidence intervals relative to zero. With a moderator, contrasts compare groups or stored moderator levels within each effect. Without a moderator, specific indirect paths are compared with one another. The displayed subtraction order is explicit.

Usage

```
plot_contrasts(
  result,
  paths = NULL,
  type = c("indirect", "paths", "overall"),
  contrasts = NULL,
  standardized = FALSE,
  engine = c("mc", "boot"),
  labels = NULL,
  title = NULL,
  x_label = NULL,
  y_label = NULL,
  base_size = 12,
  ncol = NULL
)
```

Arguments

result	A result returned by wsMed().
paths	Optional character vector of effect identifiers, in display order. NULL selects all available effects. Both indirect_1 and indirect_effect_1 are accepted when they identify an available effect.
type	"indirect" for specific indirect effects, "paths" for available conditional path coefficients, or "overall" for total indirect and total effects.
contrasts	Optional exact contrast labels to select, in display order. NULL selects all. Inspect the corresponding result table or the returned plot's data$Contrast for available labels.
standardized	Use standardized results, requested when fitting with standardized = TRUE. No scaling is recomputed from interval endpoints.
engine	For ci_method = "both", select "mc" or "boot". For a single engine its stored results are used.
labels	Optional named character vector mapping effect identifiers to display labels. Partial mappings are allowed.
title	Optional plot title.

`x_label`, `y_label`
: Optional axis labels.

`base_size`
: Base font size.

`ncol`
: Optional number of facet columns.

Details

Moderator contrasts use the existing joint-draw contrast tables. Without a moderator, only `type = "indirect"` is available: contrasts are computed from the stored joint draws, with point estimates evaluated at fitted/pooled parameters. Their intervals are percentile intervals. Standardized draws use the same draw-specific marginal endpoint scales as other standardized effects. No model is refitted. Interval endpoints for two separate effects are never subtracted to construct a contrast CI. `paths` and `labels` apply to effect panels when a moderator is present; without a moderator, `paths` selects the indirect effects to compare and `labels` maps the full contrast labels. Continuous overall contrasts are unavailable when the fitted result does not contain that contrast table.

Value

A ggplot object with plotted data and a `wsmed_plot` attribute.

See Also

`plot_effects()`, `plot_conditional_effects()`

---

plot_effects *Forest plot of mediation effects*

Description

Plot fitted indirect effects, the total indirect effect, the direct effect (`cp`) and the total effect with their stored confidence intervals. With a moderator these are effects at the model's reference value (centered continuous W = 0, or the reference category), not averages across W. Use `plot_conditional_effects()` to show selected moderator levels instead.

Usage

```
plot_effects(
  result,
  paths = NULL,
  standardized = FALSE,
  engine = c("mc", "boot"),
  labels = NULL,
  title = NULL,
  x_label = NULL,
  y_label = NULL,
  base_size = 12
)
```

Arguments

result	A result returned by wsMed().
paths	Optional character vector of effect identifiers, in display order. NULL selects all available effects. Both indirect_1 and indirect_effect_1 are accepted when they identify an available effect.
standardized	Use standardized results, requested when fitting with standardized = TRUE. No scaling is recomputed from interval endpoints.
engine	For ci_method = "both", select "mc" or "boot". For a single engine its stored results are used.
labels	Optional named character vector mapping effect identifiers to display labels. Partial mappings are allowed.
title	Optional plot title.
x_label, y_label	Optional axis labels.
base_size	Base font size.

Details

MC intervals come from the parameter draws (or the stored standardized parameter table). Bootstrap plots use the bootstrap limits, not the normal-theory limits that may also be present in the table. New workflow objects use wsmed_plots. Both interfaces expose full-precision estimate, conf.low, and conf.high columns in the plotted data; legacy column names remain available. Interval conventions are not silently changed.

Value

A ggplot object. Its data contains the plotted estimates and CI.LL/CI.UL limits. The wsmed_plot attribute records the selected engine, scale and confidence level. Use ggplot2::ggsave() to export it.

See Also

plot_conditional_effects(), plot_contrasts(), plot_moderation_curve()

---

plot_moderation_curve *Plot conditional effects across a continuous moderator*

Description

Plot a conditional indirect effect, total effect or path coefficient with its pointwise confidence band. Highlight stored grid values whose interval excludes zero. Boundary labels use raw moderator units, not sample percentiles.

Usage

```
plot_moderation_curve(
  result,
  path_name,
  title = NULL,
  x_label = "Moderator (W, raw units)",
  y_label = "Estimate",
  ns_fill = "#FEE0D2",
  sig_fill = "#C7E9C0",
  alpha_ci = 0.35,
  alpha_sig = 0.35,
  base_size = 14,
  standardized = FALSE,
  engine = c("mc", "boot")
)
```

Arguments

`result`	A result returned by `wsMed()` with a continuous moderator.
`path_name`	One stored effect identifier, such as `"indirect_effect_1_2"`, `"total_indirect"` or `"b_1_2"`. `"indirect_1_2"` is also accepted as an alias.
`title`	Optional plot title.
`x_label`, `y_label`	Axis labels. The default x axis uses raw W units.
`ns_fill`	Colour of the complete confidence band.
`sig_fill`	Colour of the regions whose pointwise CI excludes zero.
`alpha_ci`, `alpha_sig`	Opacity of the band and highlighted regions.
`base_size`	Base font size.
`standardized`	Use the stored standardized conditional results.
`engine`	For `ci_method = "both"`, select `"mc"` or `"boot"`.

Details

Highlighted intervals are approximate ranges on the stored grid, not exact Johnson–Neyman boundaries or simultaneous confidence regions. A label `[a, b]` reports the first and last qualifying grid values. An isolated qualifying point is marked by a vertical line. Missing confidence limits break a highlighted range. The displayed confidence level follows `result$alpha`; no fixed p-value threshold is implied.

Value

A ggplot object. Its `data` contains the selected curve; the `wsmed_regions` attribute contains raw-unit grid interval endpoints. The `wsmed_plot` attribute records engine, scale and confidence level.

See Also

plot_effects(), plot_conditional_effects(), plot_contrasts()

PrepareData *Prepare data for two-condition mediation*

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 scalar: one moderator column name.
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. Moderator `W` (at most one):
   - 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`, `wsmed_categorical`

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	A character vector specifying the names of moderator variables that are used to generate interaction terms with mediators. These variables will be included in the imputation model and passed to PrepareData. Default is NULL.
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
```

---

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)
```

---

standardize_moderation

Standardize conditional mediation results using existing joint draws

---

Description

Standardize conditional mediation results using existing joint draws

Usage

```
standardize_moderation(object)
```

Arguments

object	A wsMed result with a moderator and stored MC or bootstrap draws.

Details

Conditional indirect and total effects are divided by the marginal model-implied SD of Ydiff. Conditional paths use their endpoint scales. Moderator probe values remain fixed in raw units; continuous interaction coefficients in mod_coeff are reported per SD of W. Dummy variables retain 0/1 units. Coefficients and scales are transformed jointly for each draw. Intervals are percentile intervals, as in the raw conditional tables. Metadata attributes record scales, probe conventions and

draw diagnostics. With fixed.x=TRUE, external moments are held fixed. MI uses pooled primitive parameters and the existing first-imputation probing reference. For models with endogenous products, marginal variances are pooled jointly with coefficients before taking square roots, retaining their sampling covariance. Saved endogenous-product fits and MI draws require compatible algorithm identifiers. Refit unversioned old models rather than restandardizing them.

Value

A moderation list with the same conditional tables and curves as the raw output, or a list with mc and boot components for ci_method="both".

`wsMed` *Analyze two-condition within-subject mediation*

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", "UD"),
  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,
  paths = NULL
)
```

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"`. Automatic numeric 0/1 moderator detection emits a message and is recorded in `fit$diagnostics$input_notes`. Use an explicit type or factor for consistency with `wsmed_fit()`.
`MP`	Character vector identifying which regression paths are moderated (for example, `"a1"`, `"b_1_2"`, `"cp"`). Main effects of W already included in each regression are always used when computing conditional intercepts, whether or not a/cp is explicitly listed in MP.
`form`	Model type: `"P"`, `"CN"`, `"CP"`, `"PC"`, or `"UD"`. Use `"UD"` to specify a user-defined mediation model. Values are case-insensitive.
`Na`	Missing-data method: `"DE"`, `"FIML"`, or `"MI"`. Values are case-insensitive.
`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()`.
	`completed` Optional list of completed wide data frames or a `mice` mids object. Observed values, rows and factor levels must be preserved. The number of imputations is inferred; supplied m must match. Omit method_num. With external imputations, seed still controls MC only. See `wsmed_fit` for model-compatible imputation guidance.
	`decomposition` Covariance-decomposition method (`"eigen"`, `"chol"`, `"svd"`). The default `"eigen"` uses the principal symmetric covariance square root; fixed-seed MI draws differ from wsMed 1.1.0.
	`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; CI type for stored legacy bootstrap parameter tables: `"perc"`, `"bc"`, or `"bca.simple"` (zero acceleration). Current effect reports use percentile intervals regardless of this setting.

iseed, seed	Integer seeds for bootstrap and Monte Carlo, respectively.
fixed.x	Logical; passed to **lavaan**.
ci_method	CI engine: "mc", "bootstrap", or "both". Values are case-insensitive. The default (also used for NULL) is "mc". With Na = "MI", only "mc" is supported.
MCmethod	NULL or "mc". The unimplemented "bootSD" option is rejected explicitly.
standardized	Logical; if TRUE, return standardized parameter tables (including defined effects at the reference moderator value). Raw conditional results remain in moderation; standardized conditional tables and curves are added in moderation_std. Default FALSE. This choice is saved as the default for print, summary, wsmed_effects and generic plot; standardized = TRUE or FALSE on those methods overrides it for one query. Raw fitting parameters remain available through coef and vcov.
verbose	Logical; print progress messages.
paths	A character vector defining directed paths when form = "UD". Paths are specified using mediator labels M1, M2, and so on, with Y denoting the outcome. For example, c("M1 -> M3", "M3 -> Y", "M2 -> Y"). Must be NULL for the predefined model forms.

Details

Model structures:

- "P": parallel mediation
- "CN": chained (serial) mediation
- "CP": chained then parallel
- "PC": parallel then chained
- "UD": user-defined mediation model

Missing-data strategies:

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

Confidence-interval engines:

- Bootstrap: participant resampling (DE and FIML only)
- Monte Carlo: joint parameter draws via **semmcci** for DE/FIML, or the package's pooled-parameter sampler for MI

Monte Carlo inference uses the existing joint-parameter sampler. The legacy MCmethod = "bootSD" option was accepted but never applied; it now produces an explicit error rather than silently returning ordinary MC results.

Standardization divides differences by their marginal model-implied SDs, without recentering them. Dummy variables retain 0/1 units; interaction terms use the product of their component scale factors, not the interaction column SD. MC and bootstrap draws are transformed jointly with their scales. Current effect reports from print, summary and wsmed_effects, and confint applied to extracted

effects, use percentile intervals (type-7 quantiles), with or without moderation. The boot_ci_type argument controls stored legacy parameter-table intervals, not these effect reports. In standardized legacy tables, bc and bca.simple apply bias correction with zero acceleration, not full BCa. Their bootstrap p-values require at least 1000 valid replicates.

Conditional tables use percentile intervals and fixed numerical probes. With fixed.x=TRUE, scale transformations condition on the fitted external moments rather than adding sampling uncertainty for those moments. In MI, probes/centering references use the first completed data set; fixed.x=TRUE also conditions on its external moments in the pooled transform. In MI models with products of mediator differences and the moderator, marginal variances are pooled jointly with primitive coefficients, including their delta-method within-imputation and Rubin between-imputation covariance. MC propagates this joint uncertainty. Endogenous product moments are estimated even with fixed.x=TRUE.

Workflow: (1) preprocess -> (2) generate SEM syntax -> (3) fit -> (4) compute confidence intervals -> (5) optional: standardize estimates. See wsmed_categorical for supported categorical predictors, reference coding, and missing-data restrictions. In both interfaces, declare categorical covariates and moderators as unordered factors with explicit levels before fitting. The first factor level is the reference group. Existing automatic recognition of numeric 0/1 predictors is retained in this one-call interface. The empirical manuscript dataset is available as wsmed_examples; example_data is simulated tutorial data.

The one-call interface uses the same fitting and inference engines as `wsmed_model()`, `wsmed_fit()`, and `wsmed_infer()`. Default printing is concise; `print(result, detail = "full")` adds path coefficients to the structured effect report. Use `detail = "legacy"` for the original tables. Use `wsmed_effects()` to query stored draws without rerunning the analysis.

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 Raw conditional or moderated effect tables.

moderation_std Standardized conditional tables and curves when standardized=TRUE and a moderator is supplied.

form,Na,alpha Analysis settings.

paths The user-defined paths when `form = "UD"`; otherwise `NULL`.

input_vars Names of all user-supplied variables.

model,fit,inference Structured specification, reusable fit (including all imputation fits), and named stored inference objects.

Examples

```
data("example_data", package = "wsMed")
result <- wsMed(
  data = example_data,
  M_C1 = c("A1", "B1"),
```

```
  M_C2 = c("A2", "B2"),
  Y_C1 = "D1",
  Y_C2 = "D2"
)
print(result)
```

`wsmed_categorical` *Categorical predictors and supported variable types*

Description

wsMed supports binary and multicategory between-subject covariates and one categorical moderator in models with numeric outcomes and mediators.

Supported roles

Outcomes, mediators, and within-subject covariates must be numeric measures analyzed by the continuous-response SEM. Binary or ordinal response models, thresholds, and categorical links are not implemented. Storing a binary outcome as numbers does not turn the estimator into a categorical-response model. Between-subject covariates and the single moderator may instead be unordered factors. Ordered factors are rejected by `wsmed_fit()`; explicitly convert them to unordered factors only if a nominal interpretation is intended.

Declaring categories

In both the one-call and staged workflows, use `base::factor()` to declare categorical between-subject covariates and moderators in the input data. Specify levels explicitly: the first level is the reference group. For a binary predictor, `factor(x, levels = c(0, 1))` selects 0 as the reference; reversing the levels selects 1. Use unordered factors for nominal categories. Every declared factor level must have observations in the supplied data.

The existing default recognition rules are retained for compatibility. `wsmed_fit()` treats numeric predictors, including 0/1 columns, as continuous. The one-call `wsMed()` and `PrepareData()` interfaces retain automatic detection: factors, character columns, and numeric columns whose observed values are exactly 0 and 1 are categorical. Other numeric columns are continuous unless `C_type` or `W_type` overrides detection. Explicit factors provide the same categorical interpretation and reference coding in both interfaces without relying on automatic recognition. The one-call type arguments remain available to override its default rules.

Reference coding and comparisons

A factor with K levels is represented by K - 1 treatment indicators. The first factor level is the reference; `stats::relevel()` changes it before fitting. Indicators are not mean-centered. Coding does not depend on the global `contrasts` option. Between-subject indicators use `Cb*` names and moderator indicators use `W*` names. The `C_info` and `W_info` attributes of prepared data record their mappings. Labels such as `low vs high` identify the reference and comparison level: the indicator is 0 at low and 1 at high, so its coefficient is high minus low.

wsmed_effects() accepts original factor labels in at; by default it extracts every level. wsmed_contrasts() compares effects using their joint draws, preserving covariance. Do not subtract the endpoints of separate confidence intervals to obtain an interval for a difference. Marginally standardized conditional effects use the common model-implied outcome-difference SD, not separate within-category SDs; factor labels and dummy indicators retain their coding. See standardize_moderation(). Generated models account for the dependence of endogenous products on upstream mediator disturbances. Reference changes preserve the implied marginal scales of the same fitted model. MI pools marginal variance estimates jointly with coefficients; compare the same completed datasets when checking recoding. Refit saved models from before this covariance correction.

Missing values

The staged default missing = "error" rejects missing analysis values. Listwise deletion and multiple imputation are available for incomplete categorical predictors. FIML is allowed with fully observed categorical predictors but wsmed_fit() rejects missing categorical predictors; the one-call interface should follow the same restriction. MI imputes the raw variables before constructing dummy variables and interactions. The staged mi$method argument selects the method for numeric variables (default "pmm"). Factors use logistic regression for two levels and multinomial logistic regression for more than two levels. This differs from the lower-level ImputeData() method argument. MI supports Monte Carlo inference only.

See Also

wsmed_model(), wsmed_fit(), wsmed_effects(), wsmed_contrasts(), PrepareData(), ImputeData()

Examples

```
data(example_data)
dat <- example_data
dat$Group <- factor(dat$Group, levels = c("low", "med", "high"))
prepared <- PrepareData(dat, M_C1 = "A1", M_C2 = "A2",
                        Y_C1 = "D1", Y_C2 = "D2",
                        W = "Group", W_type = "categorical")
attr(prepared, "W_info")
```

---

wsmed_contrasts *Contrast effects using their joint stored draws*

Description

Contrast effects using their joint stored draws

Usage

```
wsmed_contrasts(object, contrasts, level = object$query$level)
```

Arguments

object	A wsmed_results object, containing every effect to compare.
contrasts	Named list of named numeric weight vectors. Weight names must match the term column of as.data.frame(object).
level	Confidence level; defaults to the input result's level.

Value

A wsmed_results object. Intervals are percentile intervals of the jointly evaluated contrasts, not differences between confidence limits.

---

wsmed_effects *Extract effects from fitted models or stored inference*

Description

Extract effects from fitted models or stored inference

Usage

```
wsmed_effects(
  object,
  type = c("indirect", "all", "total", "direct", "total_indirect", "paths", "parameters"),
  terms = NULL,
  at = NULL,
  scale = NULL,
  method = NULL,
  level = NULL,
  standardized = NULL
)
```

Arguments

object	A wsmed_fit, wsmed_inference, or wsMed result.
type	Effect family: indirect, all (direct, specific indirect, total indirect and total effects), total, direct, total_indirect, paths, or parameters (primitive model parameters).
terms	Optional coefficient or effect names within the selected family.
at	Named list of raw moderator values, for example list(Age = c(30, 50)). Defaults to the centering reference for a continuous moderator, or all observed levels for a categorical moderator. Parameters ignore no probes: supplying at with type = "parameters" is an error.
scale	Compatibility spelling: raw or marginal. Explicit conflicting scale and standardized arguments are rejected.

`method`	Stored inference to use when object is wsMed: mc or bootstrap. Required if both are stored. A fit alone returns point estimates only.
`level`	Confidence level; defaults to the stored inference level or .95.
`standardized`	NULL inherits the saved analysis setting (FALSE for older objects without a setting). TRUE uses model-implied marginal endpoint SDs, with or without moderation. FALSE returns raw effects.

Details

Extraction never fits, imputes, or samples. Effect and contrast intervals are percentile intervals from the stored joint draws, including when bootstrap parameter tables were requested with bias correction. For conditional standardization, moderator probes remain fixed in raw units. Scales and coefficients vary jointly across draws. Invalid standardized draws are excluded jointly and their original identities are recorded. For MI the existing first-imputation reference and per-imputation centering are retained. Endogenous products have an auxiliary moment model that retains the necessary upstream residual covariances. MI moderated models jointly pool marginal variances with coefficients, using a delta-method within-imputation covariance and Rubin's rules; MC draws retain their covariance. Refit objects saved before this correction. Finite MC draws or new imputations need not match after recoding. The query interpretation records the plug-in outcome-difference SD. Conditional result tables include `extrapolated` flags relative to observed moderator values among used participants (imputed values excluded). Out-of-range continuous probes signal `wsmed_extrapolation_warning` once per extraction; this checks the marginal range, not joint covariate overlap or data density. Categorical tables include `n.observed`, `n.fitted.min`, and `n.fitted.max`; for MI the latter span the completed datasets and are not effective sample sizes. The descriptive support metadata are retained in `query$support`. Incompatible saved fits stop before extraction. If a fit is compatible but its MI sampler is unversioned or outdated, rerun `wsmed_infer()` on that fit. Stored primitive coefficients and full legacy tables remain inspectable.

Value

A wsmed_results object with a full-precision table, joint effect draws, draw identities, query settings, and diagnostics.

`wsmed_examples` *Data for the manuscript examples*

Description

The prepared analysis data used in the three wsMed manuscript examples, derived from the iWeek study deposited by Sagaribay (2025). Each row is one participant; T1 and T2 denote measurements before and after the study period. The examples use T2 minus T1 as the within-person contrast.

Usage

```
wsmed_examples
```

Format

A data frame with 123 rows and 14 numeric columns:

TSRQ_T1, TSRQ_T2 Autonomous motivation scores at T1 and T2.

WEMBS_T1, WEMBS_T2 Subjective well-being scores at T1 and T2.

SMUF_T1, SMUF_T2 Daily social media use scores at T1 and T2.

ISI_T1, ISI_T2 Insomnia severity scores at T1 and T2.

DASSS_T1, DASSS_T2 DASS stress-subscale scores at T1 and T2, each calculated as the sum of the seven stress items.

HES_T1, HES_T2 Healthy eating scores at T1 and T2.

Age Age in years.

BMI Body mass index, recalculated from weight in pounds and height in inches using `703 * weight / height^2`.

Details

Score columns retain their prepared source values without rescaling or standardization. The original dataset contains 221 participants: 123 in the iWeek intervention group and 98 in the control group. This prepared dataset retains the 123 intervention-group participants. The wsMed examples illustrate within-person mediation using their paired T1/T2 measurements; they do not reproduce the original study's group comparison.

The manuscript authors retained the 14 analysis columns and recalculated BMI. Height values outside the preparation rule of 48–84 inches, including ambiguous units, were not converted by guesswork. BMI remains missing for 13 such records. No further participants were removed, no values were imputed, and no additional cleaning is performed when creating this package dataset. SPSS storage attributes are omitted; numerical values, row order, and missing positions match the prepared `wsMed_examples.sav` replication file.

Across these 14 columns, 55 participants (44.7 percent) have at least one missing value, with 146 missing cells in total. Missing counts for paired T1/T2 columns are TSRQ: 1/22; WEMBS: 0/22; SMUF: 8/18; ISI: 0/19; DASSS: 0/23; HES: 0/20. Age is complete and BMI has 13 missing values. Counts for an individual analysis depend on the variables it selects.

In Examples 1 and 3, TSRQ, WEMBS, and SMUF are mediators; in Example 2, WEMBS and ISI precede TSRQ. DASSS is the outcome, HES is a within-subject covariate, and BMI is a between-subject covariate. Age is a covariate in Examples 1 and 2 and the moderator in Example 3.

The source data and this prepared derivative are distributed under CC BY 4.0, separately from the package code's GPL license. Attribution and modification information are also installed in '`COPYRIGHTS`'.

Source

Sagaribay, R. (2025). *iWeek: A Week-Long Social Media Abstinence Intervention and its Impact on Well-Being, Mental Health, Body Image, and Sleep in Latina College Students*. Mendeley Data, version 1. doi:10.17632/fn9n95bsh5.1. License: https://creativecommons.org/licenses/by/4.0/.

See Also

example_data(), wsMed(), wsmed_fit()

Examples

```
data(wsmed_examples)
dim(wsmed_examples)
colSums(is.na(wsmed_examples))
sum(!complete.cases(wsmed_examples))
```

---

wsmed_fit *Fit a specified mediation model without simulation*

Description

Fit a specified mediation model without simulation

Usage

```
wsmed_fit(
  model,
  data,
  missing = c("error", "listwise", "fiml", "mi"),
  mi = list(),
  fixed.x = FALSE,
  verbose = FALSE,
  standardized = FALSE
)
```

Arguments

model	A wsmed_model object.
data	A wide data frame with one row per participant.
missing	Missing-data strategy: error, listwise, fiml, or mi.
mi	MI controls: m, method, seed, or completed. Supply completed as a list of at least two completed wide data frames or a mice mids object to use externally generated imputations. m is then inferred (or must match). Omit method and seed with completed data. Inference has its own seed.
fixed.x	Whether exogenous moments are fixed in the SEM.
verbose	Show progress messages.
standardized	Logical default for effect extraction, summaries and plots. Does not change model fitting; inference inherits this setting.

Details

This release preserves the existing per-imputation centering and first-imputation reporting reference. It does not implement MI-bootstrap. Factor outcome, mediator, and within-subject covariate columns are unsupported. See wsmed_categorical for reference coding, numeric 0/1 predictors, and restrictions on missing categorical predictors. `mi$method` specifies the method for numeric variables; factors use logistic or multinomial regression. Default imputation uses main effects and need not preserve substantive-model interactions. For moderated models, consider model-compatible external imputation (e.g., SMC-FCS) and pass its completed datasets. Supplied datasets must preserve participant order, row names, observed values and factor levels, and complete all analysis variables. Difference scores and products are rebuilt separately in every dataset, then the same Rubin/MC engine is used. Validation does not establish imputation-model compatibility. Save the external imputer's settings, convergence diagnostics and software versions separately. Internal MI with moderator interactions signals one `wsmed_mi_compatibility_warning` per fit and records it in fit diagnostics. Numeric 0/1 moderators remain continuous in this interface; a message explains this choice. For categorical predictors in either interface, use unordered factors with explicit levels; the first level is the reference group. Type messages are retained as `diagnostics$input_notes` and do not change estimates or random-number state. Diagnostics include dataset-level convergence/admissibility, covariance checks, case counts, warnings, and fitted category counts. Complete-case matrix rank is descriptive, not an identification test for FIML. Counts below five receive a descriptive small-category flag, not a universal validity cutoff. Constant transformed variables are named in pre-fit errors. Non-converged MI datasets stop pooling; known inadmissible fits cannot enter inference.

Value

A `wsmed_fit` containing all fits and pooled estimates for MI.

`wsmed_infer` *Apply uncertainty quantification to an existing fit*

Description

Apply uncertainty quantification to an existing fit

Usage

```
wsmed_infer(
  object,
  method = c("mc", "bootstrap"),
  draws = NULL,
  seed = NULL,
  level = 0.95,
  interval = "perc",
  decomposition = "eigen",
  pd = TRUE,
  tol = 1e-06
)
```

Arguments

object	A wsmed_fit, or a new wsMed one-call result.
method	mc or bootstrap. MI currently supports mc only.
draws	Number of draws; defaults to 20000 for mc and 2000 for bootstrap.
seed	Random seed, independent of the imputation seed.
level	Confidence level.
interval	Bootstrap interval type: perc, bc, or bca.simple. This controls the legacy parameter table stored in the inference backend. Effect extraction and confint methods always use percentile intervals. bca.simple uses zero acceleration, rather than a full BCa calculation.
decomposition	MI covariance factorization: eigen, chol, or svd.
pd, tol	MI covariance checks; passed to the existing sampler.

Details

Inference never reimputes or changes the supplied fit. Bootstrap necessarily fits resampled datasets. No new statistical engine is introduced. Seeded calls restore the caller's random state; seed = NULL consumes the current random stream and advances it normally. Fits with endogenous products must record compatible model and MI pooling algorithm identifiers. Older unversioned moderated fits require refitting from the original data; package version alone cannot identify their algorithms.

Value

A wsmed_inference storing draws and the original fit.

---

wsmed_inspect *Inspect a modular workflow without recomputation*

Description

Inspect a modular workflow without recomputation

Usage

```
wsmed_inspect(
  object,
  component = c("model", "syntax", "data", "fits", "diagnostics", "provenance")
)
```

Arguments

object	A wsmed_model, wsmed_fit, wsmed_inference, or wsMed object.
component	model, syntax, data, fits, diagnostics, or provenance.

Value

The requested specification, syntax, data, backend fits, or metadata.

wsmed_methods *Methods for modular wsMed objects*

Description

Methods for modular wsMed objects

Usage

```
## S3 method for class 'wsMed'
print(
  x,
  digits = 3,
  detail = c("effects", "full", "overview", "legacy"),
  type = "all",
  at = NULL,
  standardized = NULL,
  method = NULL,
  level = NULL,
  scale = NULL,
  ...
)

## S3 method for class 'wsmed_model'
print(x, ...)

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

## S3 method for class 'wsmed_inference'
print(x, ...)

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

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

## S3 method for class 'wsmed_inference'
summary(
  object,
  type = "all",
  at = NULL,
  scale = NULL,
  standardized = NULL,
  ...
)
```

```
## S3 method for class 'wsMed'
summary(
  object,
  type = "all",
  at = NULL,
  scale = NULL,
  method = NULL,
  standardized = NULL,
  ...
)

## S3 method for class 'wsmed_fit'
coef(object, ...)

## S3 method for class 'wsmed_fit'
vcov(object, ...)

## S3 method for class 'wsmed_fit'
nobs(object, ...)

## S3 method for class 'wsmed_inference'
coef(object, ...)

## S3 method for class 'wsmed_inference'
vcov(object, ...)

## S3 method for class 'wsmed_inference'
nobs(object, ...)

## S3 method for class 'wsMed'
coef(object, ...)

## S3 method for class 'wsMed'
vcov(object, ...)

## S3 method for class 'wsMed'
nobs(object, ...)

## S3 method for class 'wsmed_fit'
confint(object, parm, level = NULL, ...)

## S3 method for class 'wsmed_inference'
confint(object, parm, level = NULL, standardized = NULL, scale = NULL, ...)

## S3 method for class 'wsMed'
confint(
  object,
```

```
  parm,
  level = NULL,
  method = NULL,
  standardized = NULL,
  scale = NULL,
  ...
)

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

## S3 method for class 'wsmed_results'
summary(object, level = NULL, ...)

## S3 method for class 'wsmed_results'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)

## S3 method for class 'wsmed_results'
coef(object, ...)

## S3 method for class 'wsmed_results'
vcov(object, ...)

## S3 method for class 'wsmed_results'
confint(object, parm, level = NULL, ...)

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

## S3 method for class 'summary_wsmed'
summary(object, level = NULL, ...)
```

Arguments

x, object	A wsMed or modular workflow object.
digits	Number of displayed digits. Stored estimates are not rounded.
detail	effects (default) for core effects, full for effects and path coefficients, overview for fit status, or legacy for original stored tables.
type, at, scale, method	Passed to wsmed_effects(). Summaries default to all core effects. Select method when both inference engines are stored.
standardized	NULL inherits the saved analysis setting; TRUE or FALSE overrides it without fitting or sampling. Applies with or without moderation.
level	Confidence level. NULL uses the level saved in the selected inference or results object.
...	Additional arguments, passed to effect extraction by summary and confint methods. See wsmed_plots for plotting arguments.

`parm` Names or positions of parameters/effects to include in intervals.

`row.names, optional`
Standard data-frame conversion arguments.

Value

print methods return their input invisibly. summary returns a summary_wsmed (also a wsmed_results) for inference/one-call objects, or a summary_wsmed_fit for fits. Structured summaries contain effects and coefficients data frames, info, diagnostics and selected joint effect draws. as.data.frame, coef, confint, vcov and plot on the summary concern the selected effects; coefficients contains conditional path slopes (not all primitive parameters). Use type = 'parameters' for primitive parameters. Summary display uses printCoefmat; tables retain full precision. coef returns a named numeric vector. vcov returns the fit covariance (Rubin-pooled for MI), or the empirical covariance of stored draws for inference/results. confint recomputes percentile limits from stored draws without sampling. Its default level is the saved level, including when called on an inference or one-call object; use an explicit level to override it. On those two classes confint concerns native primitive parameters, matching coef and vcov regardless of the reporting preference (explicit scale overrides are accepted). Fit-only summaries also describe native parameters. Extract effects first to obtain indirect/conditional-effect intervals. nobs returns the sample size for each fitted dataset. as.data.frame returns a full-precision results table. plot returns a ggplot.

See Also

`wsmed_reproducibility()` for an exportable analysis manifest.

---

wsmed_model *Specify a two-condition mediation model*

Description

Specify a two-condition mediation model

Usage

```
wsmed_model(
  outcome,
  mediators,
  conditions = names(outcome),
  covariates = list(),
  structure = "parallel",
  paths = NULL,
  moderator = NULL
)
```

Arguments

outcome	Named pair of outcome columns, in condition order.
mediators	Named list of pairs of mediator columns.
conditions	Two condition names. Defaults to the outcome pair names.
covariates	List with between column names and within named pairs.
structure	One of parallel, serial, serial_parallel, parallel_serial, or custom. Mediator order matters for predefined serial structures.
paths	Directed edges using mediator role names and Y, for custom models.
moderator	NULL or a list with variable, main_effects, interactions, and optionally average_interactions. Only main_effects = "all_equations" is currently supported. Interactions name edges such as "stress -> Y".

Details

All moderator main effects are explicitly included in all difference equations. Difference-slope and average-slope interactions are separate. Categorical between-subject predictors must be unordered factors in the staged workflow. Use the same explicit factor declaration with wsMed(). Set factor levels explicitly; the first level is the reference group.

Value

A wsmed_model specification, without data or fitted estimates.

See Also

wsmed_categorical for supported variable roles and factor coding.

---

wsmed_plots *Plot mediation effects from workflow objects*

Description

Plot stored effects without fitting, imputing, or drawing new samples. The returned ggplot contains the full-precision plotted table in data.

Usage

```
## S3 method for class 'wsMed'
plot(
  x,
  type = "indirect",
  terms = NULL,
  at = NULL,
  scale = NULL,
  method = NULL,
```

```
  level = NULL,
  view = c("auto", "forest", "curve", "conditional"),
  n = 51L,
  labels = NULL,
  title = NULL,
  x_label = NULL,
  y_label = NULL,
  ncol = NULL,
  base_size = 12,
  rug = FALSE,
  standardized = NULL,
  ...
)

## S3 method for class 'wsmed_inference'
plot(
  x,
  type = "indirect",
  terms = NULL,
  at = NULL,
  scale = NULL,
  method = NULL,
  level = NULL,
  view = c("auto", "forest", "curve", "conditional"),
  n = 51L,
  labels = NULL,
  title = NULL,
  x_label = NULL,
  y_label = NULL,
  ncol = NULL,
  base_size = 12,
  rug = FALSE,
  standardized = NULL,
  ...
)

## S3 method for class 'wsmed_fit'
plot(
  x,
  type = "indirect",
  terms = NULL,
  at = NULL,
  scale = NULL,
  method = NULL,
  level = NULL,
  view = c("auto", "forest", "curve", "conditional"),
  n = 51L,
  labels = NULL,
```

```
  title = NULL,
  x_label = NULL,
  y_label = NULL,
  ncol = NULL,
  base_size = 12,
  rug = FALSE,
  standardized = NULL,
  ...
)

## S3 method for class 'wsmed_results'
plot(
  x,
  view = c("auto", "forest", "curve", "conditional"),
  terms = NULL,
  level = NULL,
  labels = NULL,
  title = NULL,
  x_label = NULL,
  y_label = NULL,
  ncol = NULL,
  base_size = 12,
  ...
)
```

Arguments

x	A wsMed, wsmed_fit, wsmed_inference, or wsmed_results object.
type	Effect family passed to wsmed_effects().
terms	Effect identifiers to select, in display order. For a results object, use either effect identifiers or complete row identifiers.
at	Named list of raw moderator values or factor labels.
scale, method	Passed to wsmed_effects(). Select method explicitly when a one-call object contains both Monte Carlo and bootstrap inference.
level	Confidence level. NULL retains the stored level. For an extracted results object, intervals can be recalculated from its stored effect draws.
view	One of auto, forest, curve, or conditional. Auto uses a curve for a continuous moderator, separate category points for a categorical moderator, and a forest plot for unmoderated effects, single probes, and contrasts.
n	Number of equally spaced probes for an automatic continuous curve (default 51). The range is the finite moderator range among cases used by the first fitted dataset. In MI this is the first completed dataset.
labels	Named character vector mapping effect identifiers to labels. Labels must distinguish effects. Defaults use mediator role names.
title, x_label, y_label	Optional title and axis labels.

ncol	Number of facet columns, or NULL.
base_size	Positive base font size.
rug	Add observed raw moderator values along a continuous curve's horizontal axis. Available for fitted/inference/one-call objects only. Uses fitted cases with an observed moderator; imputed values are excluded.
standardized	NULL inherits the analysis setting; TRUE or FALSE overrides it for this plot. Available on fit, inference and one-call objects.
...	Additional arguments are rejected to catch misspellings.

Details

Curves connect finite probes; bands are pointwise percentile intervals, not simultaneous bands or Johnson–Neyman regions. Supplied probes outside the observed fitted-case range are identified in the caption and plotted data (`extrapolated`). Group counts appear in categorical captions; for MI these are ranges across completed datasets, not effective sample sizes. Plot metadata retain `support`, including the observed range or group counts. remain in raw units. A continuous conditional point plot requires explicit `at` values. Forest plots without `at` use the usual reference value. Category and effect selections retain their supplied order. A fitted object without inference produces point estimates only, with no confidence band.

Extracted results contain only their selected effects and probes: plotting them never invents a new grid. Use `wsmed_effects()` again to change the scale, method, or probes. `wsmed_contrasts()` constructs differences from joint draws. Marginal standardization uses the same draw-specific scales as effect extraction; it does not standardize a plotted table after computing intervals.

The older `plot_effects()`, `plot_conditional_effects()`, `plot_contrasts()`, and `plot_moderation_curve()` calls remain available. For wsMed objects those helpers preserve the stored legacy tables, including requested bootstrap interval types. The plot methods here use the percentile convention of `wsmed_effects()`. Plots explicitly identify their interval source.

Value

A ggplot. Its `wsmed_plot` attribute records the method, scale, confidence level, interval convention, view, and automatic-grid range.

See Also

`wsmed_effects()`, `wsmed_contrasts()`, wsmed_methods

Examples

```
data(example_data)
model <- wsmed_model(c(before = "D1", after = "D2"),
  list(A = c(before = "A1", after = "A2")),
  moderator = list(variable = "D3", interactions = "A -> Y"))
fit <- wsmed_fit(model, example_data)
plot(fit, n = 21, title = "Conditional point estimates")
```

`wsmed_reproducibility` *Export the settings and provenance of an analysis*

Description

Export the settings and provenance of an analysis

Usage

```
wsmed_reproducibility(object, file = NULL)
```

Arguments

`object`	A wsmed_model, wsmed_fit, wsmed_inference, wsmed_results, or current wsMed object. A one-call object exports all stored inference methods.
`file`	Optional new .rds file for the report. Existing files are not overwritten.

Details

This report contains no participant-level data or simulation draws. It is an analysis manifest, not a replacement for the data, analysis script, or a dependency lockfile. It never fits, imputes, samples, or updates results. Recorded fitting versions are separated from the environment used to export. Metadata absent from an older saved object remain absent, not reconstructed from currently installed package versions. Reports for extracted results include the query and contrast definition as well as the upstream settings.

Value

A plain list of model specification, settings, reference coding, diagnostics, and recorded software versions; invisibly when written to file.

Examples

```
data(example_data)
m <- wsmed_model(c(before = "D1", after = "D2"),
  list(A = c(before = "A1", after = "A2")))
f <- wsmed_fit(m, example_data)
report <- wsmed_reproducibility(f)
report$analysis$missing
```

Index

∗ **datasets**
example_data, 2
wsmed_examples, 34
∗ **models**
wsmed_categorical, 31

as.data.frame.wsmed_results *(*wsmed_methods*)*, 39

base::factor(), *31*

coef.wsMed *(*wsmed_methods*)*, 39
coef.wsmed_fit *(*wsmed_methods*)*, 39
coef.wsmed_inference *(*wsmed_methods*)*, 39
coef.wsmed_results *(*wsmed_methods*)*, 39
confint.wsMed *(*wsmed_methods*)*, 39
confint.wsmed_fit *(*wsmed_methods*)*, 39
confint.wsmed_inference *(*wsmed_methods*)*, 39
confint.wsmed_results *(*wsmed_methods*)*, 39

data.frame, *28*

example_data, 2, *30*
example_data(), *36*

GenerateModelCN, 3
GenerateModelCN(), *6*, *9*, *11*
GenerateModelCP, 5
GenerateModelCustom, 7
GenerateModelP, 8, *22*
GenerateModelP(), *5*, *6*, *11*
GenerateModelPC, 10
ggplot2::ggsave(), *19*

ImputeData, 12, *25*
ImputeData(), *32*

lavaan::sem(), *14*, *15*

MCMI2, 13

nobs.wsMed *(*wsmed_methods*)*, 39
nobs.wsmed_fit *(*wsmed_methods*)*, 39
nobs.wsmed_inference *(*wsmed_methods*)*, 39

plot.wsMed *(*wsmed_plots*)*, 43
plot.wsmed_fit *(*wsmed_plots*)*, 43
plot.wsmed_inference *(*wsmed_plots*)*, 43
plot.wsmed_results *(*wsmed_plots*)*, 43
plot_conditional_effects, 15
plot_conditional_effects(), *18*, *19*, *21*, *46*
plot_contrasts, 17
plot_contrasts(), *16*, *19*, *21*, *46*
plot_effects, 18
plot_effects(), *16*, *18*, *21*, *46*
plot_moderation_curve, 19
plot_moderation_curve(), *19*, *46*
PrepareData, 21, *24*, *25*
PrepareData(), *4–11*, *31*, *32*
PrepareMissingData, *22*, 23
print.summary_wsmed *(*wsmed_methods*)*, 39
print.summary_wsmed_fit *(*wsmed_methods*)*, 39
print.wsMed *(*wsmed_methods*)*, 39
print.wsmed_fit *(*wsmed_methods*)*, 39
print.wsmed_inference *(*wsmed_methods*)*, 39
print.wsmed_model *(*wsmed_methods*)*, 39
print.wsmed_results *(*wsmed_methods*)*, 39
printGM, 25

semmcci::MC(), *15*
semmcci::MCStd(), *15*
standardize_moderation, 26
standardize_moderation(), *32*
stats::relevel(), *31*
summary.summary_wsmed *(*wsmed_methods*)*, 39
summary.wsMed *(*wsmed_methods*)*, 39
summary.wsmed_fit *(*wsmed_methods*)*, 39

summary.wsmed_inference (wsmed_methods), 39
summary.wsmed_results (wsmed_methods), 39

vcov.wsMed (wsmed_methods), 39
vcov.wsmed_fit (wsmed_methods), 39
vcov.wsmed_inference (wsmed_methods), 39
vcov.wsmed_results (wsmed_methods), 39

wsMed, 22, 25, 27
wsMed(), 5, 6, 9, 11, 16, 17, 19, 20, 31, 36, 43
wsmed_categorical, 3, 22, 30, 31, 37, 43
wsmed_contrasts, 32
wsmed_contrasts(), 32, 46
wsmed_effects, 33
wsmed_effects(), 30, 32, 41, 45, 46
wsmed_examples, 30, 34
wsmed_examples(), 2, 3
wsmed_fit, 28, 36
wsmed_fit(), 30–32, 36
wsmed_infer, 37
wsmed_infer(), 30, 34
wsmed_inspect, 38
wsmed_methods, 39, 46
wsmed_model, 42
wsmed_model(), 30, 32
wsmed_plots, 19, 41, 43
wsmed_reproducibility, 47
wsmed_reproducibility(), 42