

Package ‘metaDyn’

May 8, 2026

Title Multivariate Meta-Analysis of Dynamic Model Estimates

Version 1.0.1

Description Fits fixed-, random-, or mixed-effects multivariate meta-analysis models using dynamic model estimates from each individual building on and extending Lee and Gates (2023) <[doi:10.1080/00273171.2023.2229310](https://doi.org/10.1080/00273171.2023.2229310)>.

URL <https://github.com/jeksterslab/metaDyn>,
<https://jeksterslab.github.io/metaDyn/>

BugReports <https://github.com/jeksterslab/metaDyn/issues>

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.1.0), OpenMx (>= 2.22.10)

Imports Matrix, fitVARMxID (>= 1.0.2)

Suggests knitr, rmarkdown, testthat, simStateSpace, MASS, metaSEM,
expm

RoxygenNote 7.3.3.9000

NeedsCompilation no

Author Ivan Jacob Agaloos Pesigan [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0003-4818-8420>>)

Maintainer Ivan Jacob Agaloos Pesigan <r.jeksterslab@gmail.com>

Repository CRAN

Date/Publication 2026-03-24 09:10:02 UTC

Contents

| | |
|-------------------------------|---|
| coef.metadynmeta | 2 |
| confint.metadynmeta | 2 |
| extract | 3 |
| extract.metadynmeta | 4 |
| Meta | 4 |
| MetaVARMx | 9 |

| | |
|---------------------|----|
| print.metadynmeta | 15 |
| summary.metadynmeta | 16 |
| vcov.metadynmeta | 17 |

| | |
|--------------|-----------|
| Index | 18 |
|--------------|-----------|

| | |
|------------------|--|
| coef.metadynmeta | <i>Estimated Parameter Method for an Object of Class metadynmeta</i> |
|------------------|--|

Description

Estimated Parameter Method for an Object of Class metadynmeta

Usage

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

Arguments

| | |
|--------|---------------------------------|
| object | an object of class metadynmeta. |
| ... | further arguments. |

Value

Returns a vector of estimated parameters.

Author(s)

Ivan Jacob Agaloos Pesigan

| | |
|---------------------|---|
| confint.metadynmeta | <i>Confidence Intervals for the Parameter Estimates</i> |
|---------------------|---|

Description

Confidence Intervals for the Parameter Estimates

Usage

```
## S3 method for class 'metadynmeta'
confint(object, parm = NULL, level = 0.95, robust = NULL, ...)
```

Arguments

| | |
|--------|--|
| object | an object of class metadynmeta. |
| parm | a specification of which parameters are to be given confidence intervals, either a vector of numbers or a vector of names. If missing, all parameters are considered. |
| level | the confidence level required. |
| robust | Logical. If TRUE, use robust (sandwich) sampling variance-covariance matrix. If FALSE, use normal theory sampling variance-covariance matrix. If NULL, the function will check object if robust standard errors are available. |
| ... | further arguments. |

Value

Returns a matrix of confidence intervals.

Author(s)

Ivan Jacob Agaloos Pesigan

extract

Extract Generic Function

Description

A generic function for extracting elements from objects.

Usage

```
extract(object, what)
```

Arguments

| | |
|--------|-------------------|
| object | An object. |
| what | Character string. |

Value

A value determined by the specific method for the object's class.

extract.metadynmeta *Extract Method for an Object of Class metadynmeta*

Description

Extract Method for an Object of Class metadynmeta

Usage

```
## S3 method for class 'metadynmeta'
extract(object, what = NULL)
```

Arguments

| | |
|--------|--|
| object | an object of class metadynmeta. |
| what | Character string. What specific matrix to extract. If what = NULL, extract all available matrices. |

Value

Returns a list of estimates.

Author(s)

Ivan Jacob Agaloos Pesigan

Meta *Fit Multivariate Meta-Analysis*

Description

This function estimates fixed-, random-, or mixed-effects meta-analytic parameters using per-individual coefficient estimates and their sampling variance-covariance matrices. Optionally, it fits distal-outcome models in which between-person outcomes are regressed on between-person covariates and the meta-analyzed parameters/effect sizes.

Usage

```
Meta(
  y,
  v,
  x = NULL,
  z = NULL,
  random = TRUE,
  alpha_free = NULL,
```

```
alpha_values = NULL,  
alpha_lbound = NULL,  
alpha_ubound = NULL,  
tau_sqr_diag = FALSE,  
tau_sqr_d_free = NULL,  
tau_sqr_d_values = NULL,  
tau_sqr_d_lbound = NULL,  
tau_sqr_d_ubound = NULL,  
tau_sqr_l_free = NULL,  
tau_sqr_l_values = NULL,  
tau_sqr_l_lbound = NULL,  
tau_sqr_l_ubound = NULL,  
i_sqr_univariate = FALSE,  
gamma_free = NULL,  
gamma_values = NULL,  
gamma_lbound = NULL,  
gamma_ubound = NULL,  
kappa_free = NULL,  
kappa_values = NULL,  
kappa_lbound = NULL,  
kappa_ubound = NULL,  
phi_free = NULL,  
phi_values = NULL,  
phi_lbound = NULL,  
phi_ubound = NULL,  
omega_free = NULL,  
omega_values = NULL,  
omega_lbound = NULL,  
omega_ubound = NULL,  
psi_diag = FALSE,  
psi_d_free = NULL,  
psi_d_values = NULL,  
psi_d_lbound = NULL,  
psi_d_ubound = NULL,  
psi_l_free = NULL,  
psi_l_values = NULL,  
psi_l_lbound = NULL,  
psi_l_ubound = NULL,  
check_estimates = TRUE,  
robust = FALSE,  
alpha = 0.05,  
seed = NULL,  
tries_explore = 100,  
tries_local = 100,  
max_attempts = 10,  
silent = FALSE,  
ncores = NULL  
)
```

Arguments

| | |
|-------------------------------|---|
| <code>y</code> | A list. Each element of the list is a numeric vector of estimated coefficients. |
| <code>v</code> | A list. Each element of the list is a sampling variance-covariance matrix of <code>y</code> . |
| <code>x</code> | An optional list. Each element of the list is a numeric vector of covariates. |
| <code>z</code> | An optional list. Each element of the list is a numeric vector of distal outcomes. |
| <code>random</code> | Logical. If <code>random = TRUE</code> , estimates random effects. If <code>random = FALSE</code> , <code>tau_sqr</code> is a null matrix. |
| <code>alpha_free</code> | Logical vector. Optional vector of free (TRUE) parameters for alpha. |
| <code>alpha_values</code> | Numeric vector. Optional vector of starting values for alpha. |
| <code>alpha_lbound</code> | Numeric vector. Optional vector of lower bound values for alpha. |
| <code>alpha_ubound</code> | Numeric vector. Optional vector of upper bound values for alpha. |
| <code>tau_sqr_diag</code> | Logical. If <code>tau_sqr_diag = TRUE</code> , <code>tau_sqr</code> is a diagonal matrix. If <code>tau_sqr_diag = FALSE</code> , <code>tau_sqr</code> is a symmetric matrix. |
| <code>tau_sqr_d_free</code> | Logical vector indicating free/fixed status of the elements of <code>tau_sqr_d</code> . If NULL, all element of <code>tau_sqr_d</code> are free. |
| <code>tau_sqr_d_values</code> | Numeric vector with starting values for <code>tau_sqr_d</code> . If NULL, defaults to a vector of ones. |
| <code>tau_sqr_d_lbound</code> | Numeric vector with lower bounds for <code>tau_sqr_d</code> . If NULL, no lower bounds are set. |
| <code>tau_sqr_d_ubound</code> | Numeric vector with upper bounds for <code>tau_sqr_d</code> . If NULL, no upper bounds are set. |
| <code>tau_sqr_l_free</code> | Logical matrix indicating which strictly-lower-triangular elements of <code>tau_sqr_l</code> are free. Ignored if <code>tau_sqr_diag = TRUE</code> . |
| <code>tau_sqr_l_values</code> | Numeric matrix of starting values for the strictly-lower-triangular elements of <code>tau_sqr_l</code> . If NULL, defaults to a null matrix. |
| <code>tau_sqr_l_lbound</code> | Numeric matrix with lower bounds for <code>tau_sqr_l</code> . If NULL, no lower bounds are set. |
| <code>tau_sqr_l_ubound</code> | Numeric matrix with upper bounds for <code>tau_sqr_l</code> . If NULL, no upper bounds are set. |
| <code>i_sqr_univariate</code> | Logical. If <code>i_sqr_univariate = TRUE</code> , use the univariate formula for I^2 . If <code>i_sqr_univariate = FALSE</code> , use the multivariate formula for I^2 . |
| <code>gamma_free</code> | Logical matrix. Optional matrix of free (TRUE) parameters for gamma. |
| <code>gamma_values</code> | Numeric matrix. Optional matrix of starting values for gamma. |
| <code>gamma_lbound</code> | Numeric matrix. Optional matrix of lower bound values for gamma. |
| <code>gamma_ubound</code> | Numeric matrix. Optional matrix of upper bound values for gamma. |

| | |
|-----------------|--|
| kappa_free | Logical vector. Optional vector of free (TRUE) parameters for kappa. |
| kappa_values | Numeric vector. Optional vector of starting values for kappa. |
| kappa_lbound | Numeric vector. Optional vector of lower bound values for kappa. |
| kappa_ubound | Numeric vector. Optional vector of upper bound values for kappa. |
| phi_free | Logical matrix. Optional matrix of free (TRUE) parameters for phi. |
| phi_values | Numeric matrix. Optional matrix of starting values for phi. |
| phi_lbound | Numeric matrix. Optional matrix of lower bound values for phi. |
| phi_ubound | Numeric matrix. Optional matrix of upper bound values for phi. |
| omega_free | Logical matrix. Optional matrix of free (TRUE) parameters for omega. |
| omega_values | Numeric matrix. Optional matrix of starting values for omega. |
| omega_lbound | Numeric matrix. Optional matrix of lower bound values for omega. |
| omega_ubound | Numeric matrix. Optional matrix of upper bound values for omega. |
| psi_diag | Logical. If <code>psi_diag = TRUE</code> , <code>psi</code> is a diagonal matrix. If <code>psi_diag = FALSE</code> , <code>psi</code> is a symmetric matrix. |
| psi_d_free | Logical vector indicating free/fixed status of the elements of <code>psi_d</code> . If NULL, all element of <code>psi_d</code> are free. |
| psi_d_values | Numeric vector with starting values for <code>psi_d</code> . If NULL, defaults to a vector of ones. |
| psi_d_lbound | Numeric vector with lower bounds for <code>psi_d</code> . If NULL, no lower bounds are set. |
| psi_d_ubound | Numeric vector with upper bounds for <code>psi_d</code> . If NULL, no upper bounds are set. |
| psi_l_free | Logical matrix indicating which strictly-lower-triangular elements of <code>psi_l</code> are free. Ignored if <code>psi_diag = TRUE</code> . |
| psi_l_values | Numeric matrix of starting values for the strictly-lower-triangular elements of <code>psi_l</code> . If NULL, defaults to a null matrix. |
| psi_l_lbound | Numeric matrix with lower bounds for <code>psi_l</code> . If NULL, no lower bounds are set. |
| psi_l_ubound | Numeric matrix with upper bounds for <code>psi_l</code> . If NULL, no upper bounds are set. |
| check_estimates | Logical. Check elements of <code>v</code> for positive definiteness. If the test fails, the function generates a near positive definite matrix to replace the original using <code>Matrix::nearPD()</code> . |
| robust | Logical. If TRUE, calculate robust (sandwich) sampling variance-covariance matrix in stage 2. |
| alpha | Numeric. Alpha for test of significance and confidence intervals. |
| seed | Random seed for reproducibility. |
| tries_explore | Integer. Number of extra tries for the wide exploration phase. |
| tries_local | Integer. Number of extra tries for local polishing. |
| max_attempts | Integer. Maximum number of remediation attempts after the first Hessian computation fails the criteria. |
| silent | Logical. If TRUE, suppresses messages during the model fitting stage. |
| ncores | Positive integer. Number of cores to use. |

Value

Returns an object of class `metadynmeta` which is a list with the following elements:

call Function call.

args List of function arguments.

fun Function used ("Meta").

output A fitted OpenMx model.

robust Output from `OpenMx::imxRobustSE()` with argument `details = TRUE` if `robust = TRUE`.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Cheung, M. W.-L. (2015). *Meta-analysis: A structural equation modeling approach*. Wiley. doi:10.1002/9781118957813

Neale, M. C., Hunter, M. D., Pritikin, J. N., Zahery, M., Brick, T. R., Kirkpatrick, R. M., Estabrook, R., Bates, T. C., Maes, H. H., & Boker, S. M. (2015). OpenMx 2.0: Extended structural equation and statistical modeling. *Psychometrika*, 81(2), 535–549. doi:10.1007/s1133601494358

See Also

Other Meta-Analysis of VAR Functions: [MetaVARMx\(\)](#)

Examples

```
if (requireNamespace("simStateSpace")) {
  # Generate data using the simStateSpace package-----
  library(simStateSpace)
  set.seed(42)
  n <- 5
  time <- 100
  p <- 2
  alpha <- rep(x = 0, times = p)
  beta <- 0.50 * diag(p)
  psi <- 0.001 * diag(p)
  psi_l <- t(chol(psi))
  mu0 <- SSMMeanEta(
    beta = beta,
    alpha = alpha
  )
  sigma0 <- SSMCovEta(
    beta = beta,
    psi = psi
  )
  sigma0_l <- t(chol(sigma0))
  sim <- SimSSMVARFixed(
    n = n,
    time = time,
```

```

    mu0 = mu0,
    sigma0_l = sigma0_l,
    alpha = alpha,
    beta = beta,
    psi_l = psi_l
  )
  data <- as.data.frame(sim)

  # Stage 1-----
  library(fitVARMxID)
  stage1 <- FitVARMxID(
    data = data,
    observed = paste0("y", seq_len(p)),
    id = "id",
    center = TRUE
  )
  summary(stage1)
  # Stage 2-----
  # Meta-analyze set point vector and matrix of lagged-effects
  y <- coef(
    object = stage1,
    mu = TRUE,
    beta = TRUE,
    alpha = FALSE,
    nu = FALSE,
    psi = FALSE,
    theta = FALSE
  )
  v <- vcov(
    object = stage1,
    mu = TRUE,
    beta = TRUE,
    alpha = FALSE,
    nu = FALSE,
    psi = FALSE,
    theta = FALSE
  )
  library(metaDyn)
  stage2 <- Meta(y = y, v = v, random = FALSE)
  # Methods for the output of the Meta() function
  print(stage2)
  summary(stage2)
  coef(stage2)
  vcov(stage2)
  confint(stage2)
  extract(stage2, what = "alpha")
}

```

Description

This function estimates fixed-, random-, or mixed-effects meta-analytic parameters using per-individual coefficient estimates and their sampling variance-covariance matrices. Optionally, it fits distal-outcome models in which between-person outcomes are regressed on between-person covariates and the meta-analyzed parameters/effect sizes. This function uses the estimated coefficients and sampling variance-covariance matrix from each individual fitted using the `fitVARMxID::FitVARMxID()` function.

Usage

```
MetaVARMx(  
  object,  
  x = NULL,  
  z = NULL,  
  random = TRUE,  
  alpha_free = NULL,  
  alpha_values = NULL,  
  alpha_lbound = NULL,  
  alpha_ubound = NULL,  
  tau_sqr_diag = FALSE,  
  tau_sqr_d_free = NULL,  
  tau_sqr_d_values = NULL,  
  tau_sqr_d_lbound = NULL,  
  tau_sqr_d_ubound = NULL,  
  tau_sqr_l_free = NULL,  
  tau_sqr_l_values = NULL,  
  tau_sqr_l_lbound = NULL,  
  tau_sqr_l_ubound = NULL,  
  i_sqr_univariate = FALSE,  
  gamma_free = NULL,  
  gamma_values = NULL,  
  gamma_lbound = NULL,  
  gamma_ubound = NULL,  
  kappa_free = NULL,  
  kappa_values = NULL,  
  kappa_lbound = NULL,  
  kappa_ubound = NULL,  
  phi_free = NULL,  
  phi_values = NULL,  
  phi_lbound = NULL,  
  phi_ubound = NULL,  
  omega_free = NULL,  
  omega_values = NULL,  
  omega_lbound = NULL,  
  omega_ubound = NULL,  
  psi_diag = FALSE,  
  psi_d_free = NULL,  
  psi_d_values = NULL,
```

```

psi_d_lbound = NULL,
psi_d_ubound = NULL,
psi_l_free = NULL,
psi_l_values = NULL,
psi_l_lbound = NULL,
psi_l_ubound = NULL,
check_estimates = TRUE,
effects = TRUE,
set_point = TRUE,
int_meas = TRUE,
int_dyn = TRUE,
cov_meas = TRUE,
cov_dyn = TRUE,
robust_v = FALSE,
robust = FALSE,
alpha = 0.05,
seed = NULL,
tries_explore = 100,
tries_local = 100,
max_attempts = 10,
silent = FALSE,
ncores = NULL
)

```

Arguments

| | |
|------------------|--|
| object | Output of the <code>fitVARMxID::FitVARMxID()</code> function. |
| x | An optional list. Each element of the list is a numeric vector of covariates. |
| z | An optional list. Each element of the list is a numeric vector of distal outcomes. |
| random | Logical. If <code>random = TRUE</code> , estimates random effects. If <code>random = FALSE</code> , <code>tau_sqr</code> is a null matrix. |
| alpha_free | Logical vector. Optional vector of free (TRUE) parameters for alpha. |
| alpha_values | Numeric vector. Optional vector of starting values for alpha. |
| alpha_lbound | Numeric vector. Optional vector of lower bound values for alpha. |
| alpha_ubound | Numeric vector. Optional vector of upper bound values for alpha. |
| tau_sqr_diag | Logical. If <code>tau_sqr_diag = TRUE</code> , <code>tau_sqr</code> is a diagonal matrix. If <code>tau_sqr_diag = FALSE</code> , <code>tau_sqr</code> is a symmetric matrix. |
| tau_sqr_d_free | Logical vector indicating free/fixed status of the elements of <code>tau_sqr_d</code> . If NULL, all element of <code>tau_sqr_d</code> are free. |
| tau_sqr_d_values | Numeric vector with starting values for <code>tau_sqr_d</code> . If NULL, defaults to a vector of ones. |
| tau_sqr_d_lbound | Numeric vector with lower bounds for <code>tau_sqr_d</code> . If NULL, no lower bounds are set. |

| | |
|-------------------------------|---|
| <code>tau_sqr_d_ubound</code> | Numeric vector with upper bounds for <code>tau_sqr_d</code> . If NULL, no upper bounds are set. |
| <code>tau_sqr_l_free</code> | Logical matrix indicating which strictly-lower-triangular elements of <code>tau_sqr_l</code> are free. Ignored if <code>tau_sqr_diag = TRUE</code> . |
| <code>tau_sqr_l_values</code> | Numeric matrix of starting values for the strictly-lower-triangular elements of <code>tau_sqr_l</code> . If NULL, defaults to a null matrix. |
| <code>tau_sqr_l_lbound</code> | Numeric matrix with lower bounds for <code>tau_sqr_l</code> . If NULL, no lower bounds are set. |
| <code>tau_sqr_l_ubound</code> | Numeric matrix with upper bounds for <code>tau_sqr_l</code> . If NULL, no upper bounds are set. |
| <code>i_sqr_univariate</code> | Logical. If <code>i_sqr_univariate = TRUE</code> , use the univariate formula for I^2 . If <code>i_sqr_univariate = FALSE</code> , use the multivariate formula for I^2 . |
| <code>gamma_free</code> | Logical matrix. Optional matrix of free (TRUE) parameters for gamma. |
| <code>gamma_values</code> | Numeric matrix. Optional matrix of starting values for gamma. |
| <code>gamma_lbound</code> | Numeric matrix. Optional matrix of lower bound values for gamma. |
| <code>gamma_ubound</code> | Numeric matrix. Optional matrix of upper bound values for gamma. |
| <code>kappa_free</code> | Logical vector. Optional vector of free (TRUE) parameters for kappa. |
| <code>kappa_values</code> | Numeric vector. Optional vector of starting values for kappa. |
| <code>kappa_lbound</code> | Numeric vector. Optional vector of lower bound values for kappa. |
| <code>kappa_ubound</code> | Numeric vector. Optional vector of upper bound values for kappa. |
| <code>phi_free</code> | Logical matrix. Optional matrix of free (TRUE) parameters for phi. |
| <code>phi_values</code> | Numeric matrix. Optional matrix of starting values for phi. |
| <code>phi_lbound</code> | Numeric matrix. Optional matrix of lower bound values for phi. |
| <code>phi_ubound</code> | Numeric matrix. Optional matrix of upper bound values for phi. |
| <code>omega_free</code> | Logical matrix. Optional matrix of free (TRUE) parameters for omega. |
| <code>omega_values</code> | Numeric matrix. Optional matrix of starting values for omega. |
| <code>omega_lbound</code> | Numeric matrix. Optional matrix of lower bound values for omega. |
| <code>omega_ubound</code> | Numeric matrix. Optional matrix of upper bound values for omega. |
| <code>psi_diag</code> | Logical. If <code>psi_diag = TRUE</code> , <code>psi</code> is a diagonal matrix. If <code>psi_diag = FALSE</code> , <code>psi</code> is a symmetric matrix. |
| <code>psi_d_free</code> | Logical vector indicating free/fixed status of the elements of <code>psi_d</code> . If NULL, all element of <code>psi_d</code> are free. |
| <code>psi_d_values</code> | Numeric vector with starting values for <code>psi_d</code> . If NULL, defaults to a vector of ones. |
| <code>psi_d_lbound</code> | Numeric vector with lower bounds for <code>psi_d</code> . If NULL, no lower bounds are set. |
| <code>psi_d_ubound</code> | Numeric vector with upper bounds for <code>psi_d</code> . If NULL, no upper bounds are set. |

| | |
|-----------------|---|
| psi_l_free | Logical matrix indicating which strictly-lower-triangular elements of psi_l are free. Ignored if psi_diag = TRUE. |
| psi_l_values | Numeric matrix of starting values for the strictly-lower-triangular elements of psi_l. If NULL, defaults to a null matrix. |
| psi_l_lbound | Numeric matrix with lower bounds for psi_l. If NULL, no lower bounds are set. |
| psi_l_ubound | Numeric matrix with upper bounds for psi_l. If NULL, no upper bounds are set. |
| check_estimates | Logical. Check elements of v for positive definiteness. If the test fails, the function generates a near positive definite matrix to replace the original using <code>Matrix::nearPD()</code> . |
| effects | Logical. If effects = TRUE, include estimates of the dynamic effects matrix, if available. If effects = FALSE, exclude estimates of the dynamic effects matrix. |
| set_point | Logical. If set_point = TRUE, include estimates of the set-point vector, if available. If set_point = FALSE, exclude estimates of the set-point vector. |
| int_meas | Logical. If int_meas = TRUE, include estimates of the measurement intercept vector, if available. If int_meas = FALSE, exclude estimates of the measurement intercept vector. |
| int_dyn | Logical. If int_dyn = TRUE, include estimates of the dynamic process intercept vector, if available. If int_dyn = FALSE, exclude estimates of the dynamic process intercept vector. |
| cov_meas | Logical. If cov_meas = TRUE, include estimates of the measurement error covariance matrix, if available. If cov_meas = FALSE, exclude estimates of the measurement error covariance matrix. |
| cov_dyn | Logical. If cov_dyn = TRUE, include estimates of the process noise covariance matrix, if available. If cov_dyn = FALSE, exclude estimates of the process noise covariance matrix. |
| robust_v | Logical. If TRUE, use robust (sandwich) sampling variance-covariance matrix in stage 1. If FALSE, use normal theory sampling variance-covariance matrix in stage 1. |
| robust | Logical. If TRUE, calculate robust (sandwich) sampling variance-covariance matrix in stage 2. |
| alpha | NUmeric. Alpha for test of significance and confidence intervals. |
| seed | Random seed for reproducibility. |
| tries_explore | Integer. Number of extra tries for the wide exploration phase. |
| tries_local | Integer. Number of extra tries for local polishing. |
| max_attempts | Integer. Maximum number of remediation attempts after the first Hessian computation fails the criteria. |
| silent | Logical. If TRUE, suppresses messages during the model fitting stage. |
| ncores | Positive integer. Number of cores to use. |

Value

Returns an object of class `metadynmeta` which is a list with the following elements:

call Function call.

args List of function arguments.

fun Function used ("Meta").

output A fitted OpenMx model.

robust Output from `OpenMx::imxRobustSE()` with argument `details = TRUE` if `robust = TRUE`.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Cheung, M. W.-L. (2015). *Meta-analysis: A structural equation modeling approach*. Wiley. doi:10.1002/9781118957813

Neale, M. C., Hunter, M. D., Pritikin, J. N., Zahery, M., Brick, T. R., Kirkpatrick, R. M., Estabrook, R., Bates, T. C., Maes, H. H., & Boker, S. M. (2015). OpenMx 2.0: Extended structural equation and statistical modeling. *Psychometrika*, 81(2), 535–549. doi:10.1007/s1133601494358

See Also

Other Meta-Analysis of VAR Functions: [Meta\(\)](#)

Examples

```
if (requireNamespace("simStateSpace")) {
  # Generate data using the simStateSpace package-----
  library(simStateSpace)
  set.seed(42)
  n <- 5
  time <- 100
  p <- 2
  alpha <- rep(x = 0, times = p)
  beta <- 0.50 * diag(p)
  psi <- 0.001 * diag(p)
  psi_l <- t(chol(psi))
  mu0 <- SSMMeanEta(
    beta = beta,
    alpha = alpha
  )
  sigma0 <- SSMCovEta(
    beta = beta,
    psi = psi
  )
  sigma0_l <- t(chol(sigma0))
  sim <- SimSSMVARFixed(
    n = n,
    time = time,
```

```

    mu0 = mu0,
    sigma0_l = sigma0_l,
    alpha = alpha,
    beta = beta,
    psi_l = psi_l
  )
  data <- as.data.frame(sim)

  # Stage 1-----
  library(fitVARMxID)
  stage1 <- FitVARMxID(
    data = data,
    observed = paste0("y", seq_len(p)),
    id = "id",
    center = TRUE
  )
  summary(stage1)
  # Stage 2-----
  # Meta-analyze set point vector and matrix of lagged-effects
  library(metaDyn)
  stage2 <- MetaVARMx(
    object = stage1,
    random = FALSE,
    effects = TRUE,
    set_point = TRUE,
    int_meas = FALSE,
    int_dyn = FALSE,
    cov_meas = FALSE,
    cov_dyn = FALSE
  )
  # Methods for the output of the MetaVARMx() function
  print(stage2)
  summary(stage2)
  coef(stage2)
  vcov(stage2)
  confint(stage2)
  extract(stage2, what = "alpha")
}

```

```
print.metadynmeta      Print Method for Object of Class metadynmeta
```

Description

Print Method for Object of Class metadynmeta

Usage

```
## S3 method for class 'metadynmeta'
print(x, alpha = NULL, robust = NULL, digits = 4, ...)
```

Arguments

| | |
|--------|--|
| x | an object of class metadynmeta. |
| alpha | Numeric vector. Significance level α . If NULL, the function will check object for alpha used in model fitting. |
| robust | Logical. If TRUE, use robust (sandwich) sampling variance-covariance matrix. If FALSE, use normal theory sampling variance-covariance matrix. If NULL, the function will check object if robust standard errors are available. |
| digits | Integer indicating the number of decimal places to display. |
| ... | further arguments. |

Value

Returns a matrix of estimates, standard errors, test statistics, degrees of freedom, p-values, and confidence intervals.

Author(s)

Ivan Jacob Agaloos Pesigan

summary.metadynmeta *Summary Method for Object of Class metadynmeta*

Description

Summary Method for Object of Class metadynmeta

Usage

```
## S3 method for class 'metadynmeta'
summary(object, alpha = NULL, robust = NULL, digits = 4, ...)
```

Arguments

| | |
|--------|--|
| object | an object of class metadynmeta. |
| alpha | Numeric vector. Significance level α . If NULL, the function will check object for alpha used in model fitting. |
| robust | Logical. If TRUE, use robust (sandwich) sampling variance-covariance matrix. If FALSE, use normal theory sampling variance-covariance matrix. If NULL, the function will check object if robust standard errors are available. |
| digits | Integer indicating the number of decimal places to display. |
| ... | further arguments. |

Value

Returns a matrix of estimates, standard errors, test statistics, degrees of freedom, p-values, and confidence intervals.

Author(s)

Ivan Jacob Agaloos Pesigan

| | |
|------------------|---|
| vcov.metadynmeta | <i>Variance-Covariance Matrix Method for an Object of Class metadynmeta</i> |
|------------------|---|

Description

Variance-Covariance Matrix Method for an Object of Class metadynmeta

Usage

```
## S3 method for class 'metadynmeta'
vcov(object, robust = NULL, ...)
```

Arguments

| | |
|--------|--|
| object | an object of class metadynmeta. |
| robust | Logical. If TRUE, use robust (sandwich) sampling variance-covariance matrix. If FALSE, use normal theory sampling variance-covariance matrix. If NULL, the function will check object if robust standard errors are available. |
| ... | further arguments. |

Value

Returns the sampling variance-covariance matrix of the estimated parameters.

Author(s)

Ivan Jacob Agaloos Pesigan

Index

* Meta-Analysis of VAR Functions

Meta, [4](#)

MetaVARMx, [9](#)

* metaDyn

Meta, [4](#)

MetaVARMx, [9](#)

* meta

Meta, [4](#)

MetaVARMx, [9](#)

* methods

coef.metadynmeta, [2](#)

confint.metadynmeta, [2](#)

extract, [3](#)

extract.metadynmeta, [4](#)

print.metadynmeta, [15](#)

summary.metadynmeta, [16](#)

vcov.metadynmeta, [17](#)

coef.metadynmeta, [2](#)

confint.metadynmeta, [2](#)

extract, [3](#)

extract.metadynmeta, [4](#)

fitVARMxID::FitVARMxID(), [10](#), [11](#)

Matrix::nearPD(), [7](#), [13](#)

Meta, [4](#)

Meta(), [14](#)

MetaVARMx, [9](#)

MetaVARMx(), [8](#)

OpenMx::imxRobustSE(), [8](#), [14](#)

print.metadynmeta, [15](#)

summary.metadynmeta, [16](#)

vcov.metadynmeta, [17](#)