Package 'fitVARMxID'

February 27, 2026

Title Fit the Vector Autoregressive Model for Multiple Individuals

Version 1.0.2

Description Fit the vector autoregressive model for multiple individuals using the 'OpenMx' package (Hunter, 2017 <doi:10.1080/10705511.2017.1369354>).

URL https://github.com/jeksterslab/fitVARMxID, https://jeksterslab.github.io/fitVARMxID/

BugReports https://github.com/jeksterslab/fitVARMxID/issues

License MIT + file LICENSE

Encoding UTF-8

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

Imports stats, simStateSpace (>= 1.2.14)

Suggests knitr, rmarkdown, testthat

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 2026-02-27 19:50:09 UTC

Contents

`coef.varmxid` *Parameter Estimates*

Description

Parameter Estimates

Usage

```
## S3 method for class 'varmxid'
coef(
  object,
  mu = TRUE,
  alpha = TRUE,
  beta = TRUE,
  nu = TRUE,
  psi = TRUE,
  theta = TRUE,
  ncores = NULL,
  ...
)
```

Arguments

`object`	Object of class `varmxid`.
`mu`	Logical. If `mu = TRUE`, include estimates of the `mu` vector, if available. If `mu = FALSE`, exclude estimates of the `mu` vector.
`alpha`	Logical. If `alpha = TRUE`, include estimates of the `alpha` vector, if available. If `alpha = FALSE`, exclude estimates of the `alpha` vector.
`beta`	Logical. If `beta = TRUE`, include estimates of the `beta` matrix, if available. If `beta = FALSE`, exclude estimates of the `beta` matrix.
`nu`	Logical. If `nu = TRUE`, include estimates of the `nu` vector, if available. If `nu = FALSE`, exclude estimates of the `nu` vector.
`psi`	Logical. If `psi = TRUE`, include estimates of the `psi` matrix, if available. If `psi = FALSE`, exclude estimates of the `psi` matrix.
`theta`	Logical. If `theta = TRUE`, include estimates of the `theta` matrix, if available. If `theta = FALSE`, exclude estimates of the `theta` matrix.
`ncores`	Positive integer. Number of cores to use.
`...`	additional arguments.

Value

Returns a list of vectors of parameter estimates.

Author(s)

Ivan Jacob Agaloos Pesigan

`converged` *Check Model Convergence*

Description

Determines whether each fitted **OpenMx** model in a `varmxid` object meets convergence criteria based on (a) acceptable optimizer status and gradient size, (b) a positive-definite Hessian, and (c) parameters not being at their bounds.

Usage

```
converged(object, ...)

## S3 method for class 'varmxid'
converged(object, prop = FALSE, ...)
```

Arguments

`object`	A fit object.
`...`	Passed to and/or used by methods.
`prop`	Logical. If `prop = FALSE`, a named logical vector indicating, for each individual fit, whether the convergence criteria are met. If `prop = TRUE`, the proportion of cases that converged.

Value

For the `varmxid` method: If `prop = FALSE`, a named logical vector indicating, for each individual fit, whether the convergence criteria are met. If `prop = TRUE`, the proportion of cases that converged.

Author(s)

Ivan Jacob Agaloos Pesigan

FitVARMxID *Fit the First-Order Vector Autoregressive Model by ID*

Description

The function fits the first-order vector autoregressive model for each unit ID.

Usage

```
FitVARMxID(
  data,
  observed,
  id,
  time = NULL,
  ct = FALSE,
  center = TRUE,
  mu_fixed = FALSE,
  mu_free = NULL,
  mu_values = NULL,
  mu_lbound = NULL,
  mu_ubound = NULL,
  alpha_fixed = FALSE,
  alpha_free = NULL,
  alpha_values = NULL,
  alpha_lbound = NULL,
  alpha_ubound = NULL,
  beta_fixed = FALSE,
  beta_free = NULL,
  beta_values = NULL,
  beta_lbound = NULL,
  beta_ubound = NULL,
  psi_diag = FALSE,
  psi_fixed = FALSE,
  psi_d_free = NULL,
  psi_d_values = NULL,
  psi_d_lbound = NULL,
  psi_d_ubound = NULL,
  psi_d_equal = FALSE,
  psi_l_free = NULL,
  psi_l_values = NULL,
  psi_l_lbound = NULL,
  psi_l_ubound = NULL,
  nu_fixed = TRUE,
  nu_free = NULL,
  nu_values = NULL,
  nu_lbound = NULL,
  nu_ubound = NULL,
```

```
  theta_diag = TRUE,
  theta_fixed = TRUE,
  theta_d_free = NULL,
  theta_d_values = NULL,
  theta_d_lbound = NULL,
  theta_d_ubound = NULL,
  theta_d_equal = FALSE,
  theta_l_free = NULL,
  theta_l_values = NULL,
  theta_l_lbound = NULL,
  theta_l_ubound = NULL,
  mu0_fixed = TRUE,
  mu0_func = TRUE,
  mu0_free = NULL,
  mu0_values = NULL,
  mu0_lbound = NULL,
  mu0_ubound = NULL,
  sigma0_fixed = TRUE,
  sigma0_func = TRUE,
  sigma0_diag = FALSE,
  sigma0_d_free = NULL,
  sigma0_d_values = NULL,
  sigma0_d_lbound = NULL,
  sigma0_d_ubound = NULL,
  sigma0_d_equal = FALSE,
  sigma0_l_free = NULL,
  sigma0_l_values = NULL,
  sigma0_l_lbound = NULL,
  sigma0_l_ubound = NULL,
  robust = FALSE,
  seed = NULL,
  tries_explore = 100,
  tries_local = 100,
  max_attempts = 10,
  silent = FALSE,
  ncores = NULL
)
```

Arguments

data	Data frame. A data frame object of data for potentially multiple subjects that contain a column of subject ID numbers (i.e., an ID variable), and at least one column of observed values.
observed	Character vector. A vector of character strings of the names of the observed variables in the data.
id	Character string. A character string of the name of the ID variable in the data.
time	Character string. A character string of the name of the TIME variable in the data. Used when ct = TRUE.

ct	Logical. If TRUE, fit a continuous-time vector autoregressive model. If FALSE, fit a discrete-time vector autoregressive model.
center	Logical. If TRUE, use the mean-centered (mean-reverting) state equation. When center = TRUE, alpha is implied and the set-point mu is estimated. When center = FALSE, alpha is estimated and mu is implied.
mu_fixed	Logical. If TRUE, the set-point mean vector mu is fixed to mu_values. If mu_fixed = TRUE and mu_values = NULL, mu is fixed to a zero vector. If FALSE, mu is estimated.
mu_free	Logical vector indicating which elements of mu are freely estimated. If NULL, all elements are free. Ignored if mu_fixed = TRUE.
mu_values	Numeric vector of values for mu. If mu_fixed = TRUE, these are fixed values. If mu_fixed = FALSE, these are starting values. If NULL, defaults to a vector of zeros.
mu_lbound	Numeric vector of lower bounds for mu. Ignored if mu_fixed = TRUE.
mu_ubound	Numeric vector of upper bounds for mu. Ignored if mu_fixed = TRUE.
alpha_fixed	Logical. If TRUE, the dynamic model intercept vector alpha is fixed to alpha_values. If FALSE, alpha is estimated.
alpha_free	Logical vector indicating which elements of alpha are freely estimated. If NULL, all elements are free. Ignored if alpha_fixed = TRUE.
alpha_values	Numeric vector of values for alpha. If alpha_fixed = TRUE, these are fixed values. If alpha_fixed = FALSE, these are starting values. If NULL, defaults to a vector of zeros.
alpha_lbound	Numeric vector of lower bounds for alpha. Ignored if alpha_fixed = TRUE.
alpha_ubound	Numeric vector of upper bounds for alpha. Ignored if alpha_fixed = TRUE.
beta_fixed	Logical. If TRUE, the dynamic model coefficient matrix beta is fixed. If FALSE, beta is estimated.
beta_free	Logical matrix indicating which elements of beta are freely estimated. If NULL, all elements are free. Ignored if beta_fixed = TRUE.
beta_values	Numeric matrix. Values for beta. If beta_fixed = TRUE, these are fixed values; if beta_fixed = FALSE, these are starting values. If NULL, defaults to a diagonal matrix with -0.001 when ct = TRUE and 0.001 when ct = FALSE.
beta_lbound	Numeric matrix of lower bounds for beta. If NULL, defaults to -2.5. Ignored if beta_fixed = TRUE.
beta_ubound	Numeric matrix. Upper bounds for beta. Ignored if beta_fixed = TRUE. If NULL, defaults to +2.5. If NULL and ct = TRUE, diagonal upper bounds are set to -1e-05.
psi_diag	Logical. If TRUE, psi is diagonal. If FALSE, psi is symmetric.
psi_fixed	Logical. If TRUE, the process noise covariance matrix psi is fixed using psi_d_values and psi_l_values. If psi_d_values is NULL it is fixed to a zero matrix. If FALSE, psi is estimated.
psi_d_free	Logical vector indicating free/fixed status of the elements of psi_d. If NULL, all element of psi_d are free.

`psi_d_values` Numeric vector with starting values for `psi_d`. If `psi_fixed = TRUE`, these are fixed values. If `psi_fixed = FALSE`, these are starting values.

`psi_d_lbound` Numeric vector with lower bounds for `psi_d`.

`psi_d_ubound` Numeric vector with upper bounds for `psi_d`.

`psi_d_equal` Logical. When `TRUE`, all free diagonal elements of `psi_d` are constrained to be equal and estimated as a single shared parameter. Ignored if no diagonal elements are free.

`psi_l_free` Logical matrix indicating which strictly-lower-triangular elements of `psi_l` are free. If `NULL`, all element 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`.

`psi_l_lbound` Numeric matrix with lower bounds for `psi_l`.

`psi_l_ubound` Numeric matrix with upper bounds for `psi_l`.

`nu_fixed` Logical. If `TRUE`, the measurement model intercept vector `nu` is fixed to `nu_values`. If `FALSE`, `nu` is estimated.

`nu_free` Logical vector indicating which elements of `nu` are freely estimated. If `NULL`, all elements are free. Ignored if `nu_fixed = TRUE`.

`nu_values` Numeric vector of values for `nu`. If `nu_fixed = TRUE`, these are fixed values. If `nu_fixed = FALSE`, these are starting values. If `NULL`, defaults to a vector of zeros.

`nu_lbound` Numeric vector of lower bounds for `nu`. Ignored if `nu_fixed = TRUE`.

`nu_ubound` Numeric vector of upper bounds for `nu`. Ignored if `nu_fixed = TRUE`.

`theta_diag` Logical. If `TRUE`, `theta` is diagonal. If `FALSE`, `theta` is symmetric.

`theta_fixed` Logical. If `TRUE`, the measurement error covariance matrix `theta` is fixed using `theta_d_values` and `theta_l_values`. If `theta_d_values` is `NULL` it is fixed to a zero matrix. If `FALSE`, `theta` is estimated.

`theta_d_free` Logical vector indicating free/fixed status of the diagonal parameters `theta_d`. If `NULL`, all element of `theta_d` are free.

`theta_d_values` Numeric vector with starting values for `theta_d`. If `theta_fixed = TRUE`, these are fixed values. If `theta_fixed = FALSE`, these are starting values.

`theta_d_lbound` Numeric vector with lower bounds for `theta_d`.

`theta_d_ubound` Numeric vector with upper bounds for `theta_d`.

`theta_d_equal` Logical. When `TRUE`, all free diagonal elements of `theta_d` are constrained to be equal and estimated as a single shared parameter. Ignored if no diagonal elements are free.

`theta_l_free` Logical matrix indicating which strictly-lower-triangular elements of `theta_l` are free. If `NULL`, all element of `theta_l` are free. Ignored if `theta_diag = TRUE`.

`theta_l_values` Numeric matrix of starting values for the strictly-lower-triangular elements of `theta_l`.

`theta_l_lbound` Numeric matrix with lower bounds for `theta_l`.

theta_l_ubound Numeric matrix with upper bounds for theta_l.

mu0_fixed Logical. If TRUE, the initial mean vector mu0 is fixed. If mu0_fixed = TRUE and mu0_func = TRUE, mu0 is fixed to the implied stable mean vector. If mu0_fixed = TRUE and mu0_values = NULL, mu0 is fixed to a zero vector. If FALSE, mu0 is estimated.

mu0_func Logical. If TRUE and mu0_fixed = TRUE, mu0 is fixed to the implied stable mean vector.

mu0_free Logical vector indicating which elements of mu0 are free. Ignored if mu0_fixed = TRUE.

mu0_values Numeric vector of values for mu0. If mu0_fixed = TRUE, these are fixed values. If mu0_fixed = FALSE, these are starting values. If NULL, defaults to a vector of zeros. Ignored if mu0_fixed = TRUE and mu0_func = TRUE.

mu0_lbound Numeric vector of lower bounds for mu0. Ignored if mu0_fixed = TRUE.

mu0_ubound Numeric vector of upper bounds for mu0. Ignored if mu0_fixed = TRUE.

sigma0_fixed Logical. If TRUE, the initial condition covariance matrix sigma0 is fixed using sigma0_d_values and sigma0_l_values. If sigma0_fixed = TRUE and sigma0_func = TRUE, sigma0 is fixed to the implied stable covariance matrix. If sigma0_fixed = TRUE and sigma0_d_values = NULL, sigma0 is fixed to a diffused matrix.

sigma0_func Logical. If TRUE and sigma0_fixed = TRUE, sigma0 is fixed to the implied stable covariance matrix.

sigma0_diag Logical. If TRUE, sigma0 is diagonal. If FALSE, sigma0 is symmetric.

sigma0_d_free Logical vector indicating free/fixed status of the elements of sigma0_d. If NULL, all element of sigma0_d are free.

sigma0_d_values
Numeric vector with starting values for sigma0_d. If sigma0_fixed = TRUE, these are fixed values. If sigma0_fixed = FALSE, these are starting values.

sigma0_d_lbound
Numeric vector with lower bounds for sigma0_d.

sigma0_d_ubound
Numeric vector with upper bounds for sigma0_d.

sigma0_d_equal Logical. When TRUE, all free diagonal elements of sigma0_d are constrained to be equal and estimated as a single shared parameter. Ignored if no diagonal elements are free.

sigma0_l_free Logical matrix indicating which strictly-lower-triangular elements of sigma0_l are free. If NULL, all element of sigma0_l are free. Ignored if sigma0_diag = TRUE.

sigma0_l_values
Numeric matrix of starting values for the strictly-lower-triangular elements of sigma0_l.

sigma0_l_lbound
Numeric matrix with lower bounds for sigma0_l.

sigma0_l_ubound
Numeric matrix with upper bounds for sigma0_l.

robust	Logical. If TRUE, calculate robust (sandwich) sampling variance-covariance matrix.
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.

Details

Measurement Model:

By default, the measurement model is given by

$$\mathbf{y}_{i,t} = \boldsymbol{\eta}_{i,t}.$$

However, the full measurement model can be parameterized as follows

$$\mathbf{y}_{i,t} = \boldsymbol{\nu}_i + \boldsymbol{\Lambda}\boldsymbol{\eta}_{i,t} + \boldsymbol{\varepsilon}_{i,t}, \quad \text{with} \quad \boldsymbol{\varepsilon}_{i,t} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Theta}_i)$$

where $\mathbf{y}_{i,t}$, $\boldsymbol{\eta}_{i,t}$, and $\boldsymbol{\varepsilon}_{i,t}$ are random variables and $\boldsymbol{\nu}_i$, $\boldsymbol{\Lambda}$, and $\boldsymbol{\Theta}_i$ are model parameters. $\mathbf{y}_{i,t}$ represents a vector of observed random variables, $\boldsymbol{\eta}_{i,t}$ a vector of latent random variables, and $\boldsymbol{\varepsilon}_{i,t}$ a vector of random measurement errors, at time t and individual i. $\boldsymbol{\nu}_i$, denotes a vector of intercepts (fixed to a null vector by default), $\boldsymbol{\Lambda}$ a matrix of factor loadings, and $\boldsymbol{\Theta}_i$ the covariance matrix of $\boldsymbol{\varepsilon}$. In this model, $\boldsymbol{\Lambda}$ is an identity matrix and $\boldsymbol{\Theta}_i$ is a diagonal matrix.

Discrete-Time Dynamic Structure:

The dynamic structure is given by

$$\boldsymbol{\eta}_{i,t} = \boldsymbol{\alpha}_i + \boldsymbol{\beta}_i\boldsymbol{\eta}_{i,t-1} + \boldsymbol{\zeta}_{i,t}, \quad \text{with} \quad \boldsymbol{\zeta}_{i,t} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi}_i)$$

where $\boldsymbol{\eta}_{i,t}$, $\boldsymbol{\eta}_{i,t-1}$, and $\boldsymbol{\zeta}_{i,t}$ are random variables, and $\boldsymbol{\alpha}_i$, $\boldsymbol{\beta}_i$, and $\boldsymbol{\Psi}_i$ are model parameters. Here, $\boldsymbol{\eta}_{i,t}$ is a vector of latent variables at time t and individual i, $\boldsymbol{\eta}_{i,t-1}$ represents a vector of latent variables at time $t-1$ and individual i, and $\boldsymbol{\zeta}_{i,t}$ represents a vector of dynamic noise at time t and individual i. $\boldsymbol{\alpha}_i$ denotes a vector of intercepts, $\boldsymbol{\beta}_i$ a matrix of autoregression and cross regression coefficients, and $\boldsymbol{\Psi}_i$ the covariance matrix of $\boldsymbol{\zeta}_{i,t}$.

If center = TRUE, the dynamic structure is parameterized as follows

$$\boldsymbol{\eta}_{i,t} = \boldsymbol{\mu}_i + \boldsymbol{\beta}_i\left(\boldsymbol{\eta}_{i,t-1} - \boldsymbol{\mu}_i\right) + \boldsymbol{\zeta}_{i,t}$$

where $\boldsymbol{\mu}_i$ is equilibrium level of the latent state toward which the system is pulled over time.

Continuous-Time Dynamic Structure:

The continuous-time parameterization, when ct = TRUE, for the dynamic structure is given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \left(\boldsymbol{\alpha}_i + \boldsymbol{\beta}_i\boldsymbol{\eta}_{i,t-1}\right)\mathrm{d}t + \boldsymbol{\Psi}_i^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}$$

note that $\mathrm{d}\boldsymbol{W}$ is a Wiener process or Brownian motion, which represents random fluctuations.

If center = TRUE, the dynamic structure is parameterized as follows

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \boldsymbol{\beta}_i\left(\boldsymbol{\eta}_{i,t-1} - \boldsymbol{\mu}_i\right)\mathrm{d}t + \boldsymbol{\Psi}_i^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}$$

Value

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

call Function call.

args List of function arguments.

fun Function used ("FitVARMxID").

model A list of generated OpenMx models.

output A list of fitted OpenMx models.

converged A logical vector indicating converged cases.

robust A list of output from `OpenMx::imxRobustSE()` with argument `details = TRUE` for each `id` if `robust = TRUE`.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Hunter, M. D. (2017). State space modeling in an open source, modular, structural equation modeling environment. *Structural Equation Modeling: A Multidisciplinary Journal*, *25*(2), 307–324. doi:10.1080/10705511.2017.1369354

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 VAR Functions: `LDL()`, `Softplus()`

Examples

```
# 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 <- simStateSpace::SSMMeanEta(
  beta = beta,
  alpha = alpha
)
sigma0 <- simStateSpace::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)

# Fit the model----------------------------------------------------------
# center = TRUE
library(fitVARMxID)
fit <- FitVARMxID(
  data = data,
  observed = paste0("y", seq_len(p)),
  id = "id",
  center = TRUE
)
print(fit)
summary(fit)
coef(fit)
vcov(fit)
converged(fit)

# Fit the model----------------------------------------------------------
# center = FALSE
library(fitVARMxID)
fit <- FitVARMxID(
  data = data,
  observed = paste0("y", seq_len(p)),
  id = "id",
  center = FALSE
)
print(fit)
summary(fit)
coef(fit)
vcov(fit)
converged(fit)
```

Description

Performs an LDL' factorization of a symmetric positive-definite matrix X, such that

$$X = LDL',$$

where L is unit lower-triangular (ones on the diagonal) and D is diagonal.

Usage

```
LDL(x, epsilon = 1e-10)

InvLDL(s_l, uc_d)
```

Arguments

x	Numeric matrix. Assumed symmetric positive-definite (not checked). Note: LDL() may error if the implied diagonal entries of D are not strictly positive.
epsilon	Numeric. Small positive value used to replace *exactly zero* diagonal entries of x prior to factorization.
s_l	Matrix. Strictly lower-triangular part of L. In InvLDL(), only the strictly lower triangle is used (upper triangle and diagonal are ignored).
uc_d	Vector. Unconstrained vector such that Softplus(uc_d) = d, where d are the diagonal entries of D.

Details

LDL() returns both the unit lower-triangular factor L and the diagonal factor D. The strictly lower-triangular part of L is also provided for convenience. The function additionally computes an unconstrained vector uc_d such that Softplus(uc_d) = d. This uses a numerically stable inverse softplus implementation based on log(expm1(d)) (and a large-d rewrite), rather than the unstable expression $\log(\exp(d) - 1)$.

InvLDL() returns a symmetric positive definite matrix from the strictly lower-triangular part of L and the unconstrained vector uc_d. The reconstructed matrix is symmetrized as $(\Sigma + \Sigma')/2$ to reduce numerical asymmetry.

Value

- LDL(): a list with components:
  - l: a unit lower-triangular matrix L
  - s_l: a strictly lower-triangular part of L
  - d: a vector of diagonal entries of D
  - uc_d: unconstrained vector with $\mathrm{softplus}(uc\_d) = d$
  - x: input matrix (with diagonal zeros possibly replaced by epsilon)
  - epsilon: the epsilon value used
- InvLDL(): a symmetric positive definite matrix

See Also

Other VAR Functions: `FitVARMxID()`, `Softplus()`

Examples

```
set.seed(123)
x <- crossprod(matrix(rnorm(16), 4, 4)) + diag(1e-6, 4)
ldl <- LDL(x = x)
ldl
inv_ldl <- InvLDL(s_l = ldl$s_l, uc_d = ldl$uc_d)
inv_ldl
max(abs(x - inv_ldl))
```

`print.varmxid` *Print Method for Object of Class* `varmxid`

Description

Print Method for Object of Class `varmxid`

Usage

```
## S3 method for class 'varmxid'
print(
  x,
  means = FALSE,
  mu = TRUE,
  alpha = TRUE,
  beta = TRUE,
  nu = TRUE,
  psi = TRUE,
  theta = TRUE,
  digits = 4,
  ...
)
```

Arguments

`x`	an object of class `varmxid`.
`means`	Logical. If `means = TRUE`, return means. Otherwise, the function returns raw estimates.
`mu`	Logical. If `mu = TRUE`, include estimates of the `mu` vector, if available. If `mu = FALSE`, exclude estimates of the `mu` vector.
`alpha`	Logical. If `alpha = TRUE`, include estimates of the `alpha` vector, if available. If `alpha = FALSE`, exclude estimates of the `alpha` vector.

beta	Logical. If beta = TRUE, include estimates of the beta matrix, if available. If beta = FALSE, exclude estimates of the beta matrix.
nu	Logical. If nu = TRUE, include estimates of the nu vector, if available. If nu = FALSE, exclude estimates of the nu vector.
psi	Logical. If psi = TRUE, include estimates of the psi matrix, if available. If psi = FALSE, exclude estimates of the psi matrix.
theta	Logical. If theta = TRUE, include estimates of the theta matrix, if available. If theta = FALSE, exclude estimates of the theta matrix.
digits	Integer indicating the number of decimal places to display.
...	further arguments.

Value

Prints means or raw estimates depending the the value of the argument means.

Author(s)

Ivan Jacob Agaloos Pesigan

Softplus *Softplus and Inverse Softplus Transformations*

Description

The softplus transformation maps unconstrained real values to the positive real line. This is useful when parameters (e.g., variances) must be positive. The inverse softplus transformation recovers the unconstrained value from a strictly positive input.

Usage

```
Softplus(x)

InvSoftplus(x)
```

Arguments

x	Numeric vector or matrix. Input values to be transformed.

Details

Mathematical definitions:

- Softplus(x) = log(1 + exp(x))
- InvSoftplus(x) = log(exp(x) - 1)

Numerical implementation (stable forms):

- `Softplus(x)` = `max(x, 0) + log1p(exp(-abs(x)))`
- `InvSoftplus(y)` uses `log(expm1(y))`, and for large `y` uses the rewrite `y + log1p(-exp(-y))` for stability.

For numerical stability, these functions use `log1p()` and `expm1()` internally. `InvSoftplus()` requires strictly positive input and will error if any values are `<= 0`.

Value

- `Softplus()`: numeric vector or matrix of nonnegative values (mathematically strictly positive for finite inputs, but can underflow to 0 for very negative values).
- `InvSoftplus()`: numeric vector or matrix of unconstrained values.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other VAR Functions: `FitVARMxID()`, `LDL()`

Examples

```
# Apply softplus to unconstrained values
x <- c(-5, 0, 5)
y <- Softplus(x)

# Recover unconstrained values
x_recovered <- InvSoftplus(y)

y
x_recovered
```

---

summary.varmxid *Summary Method for Object of Class* `varmxid`

Description

Summary Method for Object of Class `varmxid`

Usage

```
## S3 method for class 'varmxid'
summary(
  object,
  means = FALSE,
  mu = TRUE,
```

```
  alpha = TRUE,
  beta = TRUE,
  nu = TRUE,
  psi = TRUE,
  theta = TRUE,
  digits = 4,
  ncores = NULL,
  ...
)
```

Arguments

object	an object of class varmxid.
means	Logical. If means = TRUE, return means. Otherwise, the function returns raw estimates.
mu	Logical. If mu = TRUE, include estimates of the mu vector, if available. If mu = FALSE, exclude estimates of the mu vector.
alpha	Logical. If alpha = TRUE, include estimates of the alpha vector, if available. If alpha = FALSE, exclude estimates of the alpha vector.
beta	Logical. If beta = TRUE, include estimates of the beta matrix, if available. If beta = FALSE, exclude estimates of the beta matrix.
nu	Logical. If nu = TRUE, include estimates of the nu vector, if available. If nu = FALSE, exclude estimates of the nu vector.
psi	Logical. If psi = TRUE, include estimates of the psi matrix, if available. If psi = FALSE, exclude estimates of the psi matrix.
theta	Logical. If theta = TRUE, include estimates of the theta matrix, if available. If theta = FALSE, exclude estimates of the theta matrix.
digits	Integer indicating the number of decimal places to display.
ncores	Positive integer. Number of cores to use.
...	further arguments.

Value

Returns means or raw estimates depending the the value of the argument means.

Author(s)

Ivan Jacob Agaloos Pesigan

vcov.varmxid *Sampling Covariance Matrix of the Parameter Estimates*

Description

Sampling Covariance Matrix of the Parameter Estimates

Usage

```
## S3 method for class 'varmxid'
vcov(
  object,
  mu = TRUE,
  alpha = TRUE,
  beta = TRUE,
  nu = TRUE,
  psi = TRUE,
  theta = TRUE,
  robust = FALSE,
  ncores = NULL,
  ...
)
```

Arguments

object	Object of class varmxid.
mu	Logical. If mu = TRUE, include estimates of the mu vector, if available. If mu = FALSE, exclude estimates of the mu vector.
alpha	Logical. If alpha = TRUE, include estimates of the alpha vector, if available. If alpha = FALSE, exclude estimates of the alpha vector.
beta	Logical. If beta = TRUE, include estimates of the beta matrix, if available. If beta = FALSE, exclude estimates of the beta matrix.
nu	Logical. If nu = TRUE, include estimates of the nu vector, if available. If nu = FALSE, exclude estimates of the nu vector.
psi	Logical. If psi = TRUE, include estimates of the psi matrix, if available. If psi = FALSE, exclude estimates of the psi matrix.
theta	Logical. If theta = TRUE, include estimates of the theta matrix, if available. If theta = FALSE, exclude estimates of the theta matrix.
robust	Logical. If TRUE, use robust (sandwich) sampling variance-covariance matrix. If FALSE, use normal theory sampling variance-covariance matrix.
ncores	Positive integer. Number of cores to use.
...	additional arguments.

Value

Returns a list of sampling variance-covariance matrices.

Author(s)

Ivan Jacob Agaloos Pesigan

Index