Package ‘fitDTVARMxID’

January 2, 2026

Title Fit the Discrete-Time Vector Autoregressive Model for Multiple Individuals

Version 0.0.0.9003

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

URL `https://github.com/jeksterslab/fitDTVARMxID`, `https://jeksterslab.github.io/fitDTVARMxID/`

BugReports `https://github.com/jeksterslab/fitDTVARMxID/issues`

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Depends R (>= 4.5.0), OpenMx (>= 2.22.10)

Imports stats, methods, Matrix

Suggests knitr, rmarkdown, testthat, simStateSpace

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`>

Contents

`coef.dtvarmxid` *Parameter Estimates*

Description

Parameter Estimates

Usage

```
## S3 method for class 'dtvarmxid'
coef(
  object,
  alpha = TRUE,
  beta = TRUE,
  nu = TRUE,
  psi = TRUE,
  theta = TRUE,
  converged = TRUE,
  grad_tol = 0.01,
  hess_tol = 1e-08,
  vanishing_theta = TRUE,
  theta_tol = 0.001,
  ...
)
```

Arguments

`object` Object of class `dtvarmxid`.

`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.

`converged` Logical. Only include converged cases.

`grad_tol` Numeric scalar. Tolerance for the maximum absolute gradient if `converged` = `TRUE`.

`hess_tol` Numeric scalar. Tolerance for Hessian eigenvalues; eigenvalues must be strictly greater than this value if `converged` = `TRUE`.

`vanishing_theta`
: Logical. Test for measurement error variance going to zero if `converged = TRUE`.

`theta_tol`
: Numeric. Tolerance for vanishing theta test if `converged` and `theta_tol` are `TRUE`.

`...`
: additional arguments.

Value

Returns a list of vectors of parameter estimates.

Author(s)

Ivan Jacob Agaloos Pesigan

`converged` *Check Model Convergence*

Description

Evaluate whether OpenMx fit has converged successfully.

Usage

```
converged(object, ...)

## S3 method for class 'dtvarmxid'
converged(
  object,
  grad_tol = 0.01,
  hess_tol = 1e-08,
  vanishing_theta = TRUE,
  theta_tol = 0.001,
  prop = FALSE,
  ...
)
```

Arguments

`object`
: An object of class `dtvarmxid`.

`...`
: Passed to and/or used by methods.

`grad_tol`
: Numeric scalar. Tolerance for the maximum absolute gradient.

`hess_tol`
: Numeric scalar. Tolerance for Hessian eigenvalues; eigenvalues must be strictly greater than this value.

`vanishing_theta`
: Logical. Test for measurement error variance going to zero.

`theta_tol` Numeric. Tolerance for vanishing theta test.

`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.

Details

Convergence is defined by three criteria:

1. Status code equals `0L`.
2. The maximum absolute gradient is below `grad_tol`.
3. The Hessian is positive definite with all eigenvalues greater than `hess_tol`.
4. If `vanishing_theta = TRUE`, the model additionally checks that the diagonal elements of the measurement error covariance matrix (Θ) are not vanishingly small, where “small” is defined by `theta_tol`.

Value

For the `dtvarmxid` 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

`FitDTVARMxID` *Fit the First-Order Discrete-Time Vector Autoregressive Model by ID*

Description

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

Usage

```
FitDTVARMxID(
  data,
  observed,
  id,
  alpha_fixed = TRUE,
  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_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,
  nu_fixed = FALSE,
  nu_free = NULL,
  nu_values = NULL,
  nu_lbound = NULL,
  nu_ubound = NULL,
  theta_diag = TRUE,
  theta_fixed = FALSE,
  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 = FALSE,
  mu0_free = NULL,
  mu0_values = NULL,
  mu0_lbound = NULL,
  mu0_ubound = NULL,
  sigma0_fixed = TRUE,
  sigma0_func = FALSE,
  sigma0_diag = FALSE,
  sigma0_d_free = NULL,
  sigma0_d_values = NULL,
  sigma0_d_lbound = NULL,
  sigma0_d_ubound = NULL,
  sigma0_l_free = NULL,
  sigma0_l_values = NULL,
  sigma0_l_lbound = NULL,
  sigma0_l_ubound = NULL,
  robust = FALSE,
  tries_explore = 100,
  tries_local = 10,
  max_attempts = 10,
  grad_tol = 0.01,
  hess_tol = 1e-08,
  eps = 1e-06,
  factor = 10,
  overwrite = FALSE,
  path = getwd(),
  prefix = "FitDTVARMxID",
  seed = NULL,
  silent = FALSE,
  ncores = NULL,
  clean = TRUE
)
```

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.

`alpha_fixed` Logical. If `TRUE`, the dynamic model intercept vector `alpha` is fixed. 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`. If `NULL`, no lower bounds are set. Ignored if `alpha_fixed = TRUE`.

`alpha_ubound` Numeric vector of upper bounds for `alpha`. If `NULL`, no upper bounds are set. 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 of values for `beta`. If `beta_fixed = TRUE`, these are fixed values. If `beta_fixed = FALSE`, these are starting values. If `NULL`, defaults to a zero matrix.

`beta_lbound` Numeric matrix of lower bounds for `beta`. If `NULL`, defaults to -1.5. Ignored if `beta_fixed = TRUE`.

`beta_ubound` Numeric matrix of upper bounds for `beta`. If `NULL`, defaults to +1.5. Ignored if `beta_fixed = TRUE`.

`psi_diag` Logical. If `TRUE`, `psi` is diagonal. If `FALSE`, `psi` is symmetric.

`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 `NULL`, defaults to a vector of ones.

`psi_d_lbound` Numeric vector with lower bounds for `psi_d`. If `NULL`, no lower bounds are set.

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

`nu_fixed` Logical. If `TRUE`, the measurement model intercept vector `nu` is fixed. 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`. If `NULL`, no lower bounds are set. Ignored if `nu_fixed = TRUE`.

`nu_ubound` Numeric vector of upper bounds for `nu`. If `NULL`, no upper bounds are set. 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 matrix `theta` is fixed to `SoftPlus(theta_d_values)`. If `FALSE`, only diagonal elements are estimated (off-diagonals fixed to zero).

`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. If `NULL`, defaults to an identity matrix.

`theta_d_lbound` Numeric vector with lower bounds for `theta_d`. If `NULL`, no lower bounds are set.

`theta_d_ubound` Numeric vector with upper bounds for `theta_d`. If `NULL`, no upper bounds are set.

`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 (`theta_eq`). Ignored if no diagonal elements are free.

`theta_l_free` Logical matrix indicating which strictly-lower-triangular elements 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`. If `NULL`, defaults to a null matrix.

`theta_l_lbound` Numeric matrix with lower bounds for `theta_l`. If `NULL`, no lower bounds are set.

theta_l_ubound Numeric matrix with upper bounds for theta_l. If NULL, no upper bounds are set.

mu0_fixed Logical. If TRUE, the initial mean vector mu0 is fixed. If FALSE, mu0 is estimated.

mu0_func Logical. If TRUE and mu0_fixed = TRUE, mu0 is fixed to $(I - \beta)^{-1}\alpha$.

mu0_free Logical vector indicating which elements of mu0 are freely estimated.

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.

mu0_lbound Numeric vector of lower bounds for mu0. If NULL, no lower bounds are set. Ignored if mu0_fixed = TRUE.

mu0_ubound Numeric vector of upper bounds for mu0. If NULL, no upper bounds are set. Ignored if mu0_fixed = TRUE.

sigma0_fixed Logical. If TRUE, the initial covariance matrix sigma0 is fixed. If FALSE, sigma0 is estimated.

sigma0_func Logical. If TRUE and sigma0_fixed = TRUE, sigma0 is fixed to $(I - \beta \otimes \beta)^{-1}\mathrm{Vec}(\Psi)$.

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 NULL, defaults to a vector of ones.

sigma0_d_lbound Numeric vector with lower bounds for sigma0_d. If NULL, no lower bounds are set.

sigma0_d_ubound Numeric vector with upper bounds for sigma0_d. If NULL, no upper bounds are set.

sigma0_l_free Logical matrix indicating which strictly-lower-triangular elements 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. If NULL, defaults to a null matrix.

sigma0_l_lbound Numeric matrix with lower bounds for sigma0_l. If NULL, no lower bounds are set.

sigma0_l_ubound Numeric matrix with upper bounds for sigma0_l. If NULL, no upper bounds are set.

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

tries_explore Integer. Number of extra tries for the wide exploration phase using OpenMx::mxTryHardWideSearch() with checkHess = FALSE.

tries_local Integer. Number of extra tries for local polishing via OpenMx::mxTryHard() when gradients remain above tolerance.

`max_attempts` Integer. Maximum number of remediation attempts after the first Hessian computation fails the criteria. Each attempt may nudge off bounds, refit locally without the Hessian, and, on the last attempt, relax bounds.

`grad_tol` Numeric. Tolerance for the maximum absolute gradient. Smaller values are stricter.

`hess_tol` Numeric. Minimum allowable Hessian eigenvalue. Smaller values are less strict.

`eps` Numeric. Proximity threshold to detect parameters on their bounds and to nudge them inward by `10 * eps`.

`factor` Numeric. Multiplicative factor to relax parameter bounds on the final remediation attempt. Lower bounds are divided by `factor` and upper bounds are multiplied by `factor`.

`overwrite` Logical. If `TRUE`, existing intermediate files are overwritten. Defaults to `FALSE`.

`path` Character string. Directory in which to save intermediate files.

`prefix` Alphanumeric character string. Prefix to use when naming intermediate files.

`seed` Random seed for reproducibility.

`silent` Logical. If `TRUE`, suppresses messages during the model fitting stage.

`ncores` Positive integer. Number of cores to use.

`clean` Logical. If `TRUE`, clean intermediate files saved in `path`.

Details

The measurement model is given by

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

where $\mathbf{y}_{i,t}$, $\boldsymbol{\eta}_{i,t}$, and $\boldsymbol{\varepsilon}_{i,t}$ are random variables and $\boldsymbol{\nu}$, $\boldsymbol{\Lambda}$, and $\boldsymbol{\Theta}$ 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. $\boldsymbol{\Lambda}$ denotes a matrix of factor loadings, and $\boldsymbol{\Theta}$ the covariance matrix of $\boldsymbol{\varepsilon}$. In this model, $\boldsymbol{\Lambda}$ is an identity matrix and $\boldsymbol{\Theta}$ is a diagonal matrix.

The dynamic structure is given by

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

where $\boldsymbol{\eta}_{i,t}$, $\boldsymbol{\eta}_{i,t-1}$, and $\boldsymbol{\zeta}_{i,t}$ are random variables, and $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\Psi}$ 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. $\boldsymbol{\alpha}$ denotes a vector of intercepts, $\boldsymbol{\beta}$ a matrix of autoregression and cross regression coefficients, and $\boldsymbol{\Psi}$ the covariance matrix of $\boldsymbol{\zeta}_{i,t}$.

Value

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

call Function call.

args List of function arguments.

fun Function used ("FitDTVARMxID").

output A list of fitted OpenMx models.

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

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

Examples

```
## Not run:
# Generate data using the simStateSpace package------------------------------
set.seed(42)
k <- 2
n <- 5
time <- 100
alpha <- rep(x = 0, times = k)
beta <- matrix(
  data = c(.5, .0, .2, .5),
  nrow = k,
  ncol = k
)
psi <- matrix(
  data = c(exp(-4.1), exp(-3.9), exp(-3.9), exp(-3.2)),
  nrow = k,
  ncol = k
)
psi_l <- t(chol(psi))
nu <- rep(x = 5, times = k)
lambda <- diag(k)
theta <- matrix(
  data = c(exp(-2), 0, 0, exp(-2.8)),
  nrow = k,
  ncol = k
)
```

```
theta_l <- t(chol(theta))
mu0 <- c(solve(diag(k) - beta) %*% alpha)
sigma0 <- matrix(
  data = c(
    solve(diag(k * k) - beta %x% beta) %*% c(psi)
  ),
  nrow = k,
  ncol = k
)
sigma0_l <- t(chol(sigma0))
sim <- simStateSpace::SimSSMIVary(
  n = n,
  time,
  mu0 = list(mu0),
  sigma0_l = list(sigma0_l),
  alpha = list(alpha),
  beta = simStateSpace::SimBetaN(
    n = n,
    beta = beta,
    vcov_beta_vec_l = t(chol(0.1 * diag(k * k)))
  ),
  psi_l = list(psi_l),
  nu = list(nu),
  lambda = list(lambda),
  theta_l = list(theta_l)
)
data <- as.data.frame(sim)

# Fit the model---------------------------------------------------------------
library(fitDTVARMxID)
fit <- FitDTVARMxID(
  data = data,
  observed = paste0(”y”, seq_len(k)),
  id = ”id”
)
print(fit)
summary(fit)
coef(fit)
vcov(fit)
converged(fit)

## End(Not run)
```

LDL *LDL’ Decomposition of a Symmetric Positive-Definite Matrix*

Description

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

$$X = LDL^{\top},$$

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

Usage

```
LDL(x)
```

Arguments

x	Numeric matrix. Must be symmetric positive-definite.

Details

This function 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 `d_uc` such that `softplus(d_uc) = d_vec`, using $\mathrm{softplus}^{-1}(y) = \log(\exp(y) - 1)$ for stable back-transformation.

Value

A list with components:

l_mat_unit	Unit lower-triangular matrix L.
l_mat_strict	Strictly lower-triangular part of L.
d_mat	Diagonal matrix D.
d_vec	Vector of diagonal entries of D.
d_uc	Unconstrained vector with $\mathrm{softplus}(d\_uc) = d\_vec$.
x	Original input matrix.
y	Reconstructed matrix $LDL^\top$.
diff	Difference `x - y`.

See Also

Other DTVAR Functions: `FitDTVARMxID()`, `Softplus()`

Examples

```
set.seed(123)
A <- matrix(rnorm(16), 4, 4)
S <- crossprod(A) + diag(1e-6, 4) # SPD
out <- LDL(S)
max(abs(out$diff))
```

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

Description

Print Method for Object of Class `dtvarmxid`

Usage

```
## S3 method for class 'dtvarmxid'
print(
  x,
  means = FALSE,
  alpha = TRUE,
  beta = TRUE,
  nu = TRUE,
  psi = TRUE,
  theta = TRUE,
  converged = TRUE,
  grad_tol = 0.01,
  hess_tol = 1e-08,
  vanishing_theta = TRUE,
  theta_tol = 0.001,
  digits = 4,
  ...
)
```

Arguments

`x` an object of class `dtvarmxid`.

`means` Logical. If `means = TRUE`, return means. Otherwise, the function returns raw estimates.

`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.

`converged` Logical. Only include converged cases.

`grad_tol` Numeric scalar. Tolerance for the maximum absolute gradient if `converged = TRUE`.

`hess_tol`	Numeric scalar. Tolerance for Hessian eigenvalues; eigenvalues must be strictly greater than this value if `converged = TRUE`.
`vanishing_theta`	Logical. Test for measurement error variance going to zero if `converged = TRUE`.
`theta_tol`	Numeric. Tolerance for vanishing theta test if `converged` and `theta_tol` are `TRUE`.
`digits`	Integer indicating the number of decimal places to display.
`...`	further arguments.

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 strictly positive. The inverse softplus transformation recovers the unconstrained value from a positive input.

Usage

```
Softplus(x)

InvSoftplus(x)
```

Arguments

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

Details

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

For numerical stability, these functions use `log1p()` and `expm1()` internally.

Value

- `Softplus()`: numeric vector or matrix of strictly positive values.
- `InvSoftplus()`: numeric vector or matrix of unconstrained values.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other DTVAR Functions: `FitDTVARMxID()`, `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.dtvarmxid` *Summary Method for Object of Class* `dtvarmxid`

Description

Summary Method for Object of Class `dtvarmxid`

Usage

```
## S3 method for class 'dtvarmxid'
summary(
  object,
  means = FALSE,
  alpha = TRUE,
  beta = TRUE,
  nu = TRUE,
  psi = TRUE,
  theta = TRUE,
  converged = TRUE,
  grad_tol = 0.01,
  hess_tol = 1e-08,
  vanishing_theta = TRUE,
  theta_tol = 0.001,
  digits = 4,
  ...
)
```

Arguments

`object` an object of class `dtvarmxid`.

`means` Logical. If `means = TRUE`, return means. Otherwise, the function returns raw estimates.

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.

converged — Logical. Only include converged cases.

grad_tol — Numeric scalar. Tolerance for the maximum absolute gradient if converged = TRUE.

hess_tol — Numeric scalar. Tolerance for Hessian eigenvalues; eigenvalues must be strictly greater than this value if converged = TRUE.

vanishing_theta — Logical. Test for measurement error variance going to zero if converged = TRUE.

theta_tol — Numeric. Tolerance for vanishing theta test if converged and theta_tol are TRUE.

digits — Integer indicating the number of decimal places to display.

... — further arguments.

Author(s)

Ivan Jacob Agaloos Pesigan

---

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

Description

Sampling Covariance Matrix of the Parameter Estimates

Usage

```
## S3 method for class 'dtvarmxid'
vcov(
  object,
  alpha = TRUE,
  beta = TRUE,
  nu = TRUE,
  psi = TRUE,
  theta = TRUE,
  converged = TRUE,
  grad_tol = 0.01,
  hess_tol = 1e-08,
  vanishing_theta = TRUE,
  theta_tol = 0.001,
  robust = FALSE,
  ...
)
```

Arguments

`object` Object of class `dtvarmxid`.

`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.

`converged` Logical. Only include converged cases.

`grad_tol` Numeric scalar. Tolerance for the maximum absolute gradient if `converged = TRUE`.

`hess_tol` Numeric scalar. Tolerance for Hessian eigenvalues; eigenvalues must be strictly greater than this value if `converged = TRUE`.

`vanishing_theta`
Logical. Test for measurement error variance going to zero if `converged = TRUE`.

`theta_tol` Numeric. Tolerance for vanishing theta test if `converged` and `theta_tol` are `TRUE`.

`robust` Logical. If `TRUE`, use robust (sandwich) sampling variance-covariance matrix. If `FALSE`, use normal theory sampling variance-covariance matrix.

`...` additional arguments.

Value

Returns a list of sampling variance-covariance matrices.

Author(s)

Ivan Jacob Agaloos Pesigan

Index