Package ‘simStateSpace’

January 8, 2025

Title Simulate Data from State Space Models

Version 1.2.5

Description Provides a streamlined and user-friendly framework
for simulating data in state space models,
particularly when the number of subjects/units (n) exceeds one,
a scenario commonly encountered in social and behavioral sciences.
For an introduction to state space models in social and behavioral sciences,
refer to Chow, Ho, Hamaker, and Dolan (2010) <doi:10.1080/10705511003661553>.

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

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

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 3.5.0)

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, stats, dynr

Suggests knitr, rmarkdown, testthat, expm

SystemRequirements GSL (>= 2.6)

RoxygenNote 7.3.2

NeedsCompilation yes

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

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

Contents

`as.data.frame.simstatespace`

Coerce an Object of Class `simstatespace` *to a Data Frame*

Description

Coerce an Object of Class `simstatespace` to a Data Frame

Usage

```
## S3 method for class 'simstatespace'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  eta = FALSE,
```

```
    long = TRUE,
    ...
)
```

Arguments

x	Object of class `simstatespace`.
row.names	`NULL` or character vector giving the row names for the data frame. Missing values are not allowed.
optional	Logical. If `TRUE`, setting row names and converting column names is optional.
eta	Logical. If `eta = TRUE`, include `eta`. If `eta = FALSE`, exclude `eta`.
long	Logical. If `long = TRUE`, use long format. If `long = FALSE`, use wide format.
...	Additional arguments.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
# prepare parameters
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
## dynamic structure
p <- 3
mu0 <- rep(x = 0, times = p)
sigma0 <- diag(p)
sigma0_l <- t(chol(sigma0))
alpha <- rep(x = 0, times = p)
beta <- 0.50 * diag(p)
psi <- diag(p)
psi_l <- t(chol(psi))
## measurement model
k <- 3
nu <- rep(x = 0, times = k)
lambda <- diag(k)
theta <- 0.50 * diag(k)
theta_l <- t(chol(theta))
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
```

```
      nrow = j,
      ncol = time
    )
  }
)
gamma <- diag(x = 0.10, nrow = p, ncol = j)
kappa <- diag(x = 0.10, nrow = k, ncol = j)

# Type 0
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)

head(as.data.frame(ssm))
head(as.data.frame(ssm, long = FALSE))

# Type 1
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
)

head(as.data.frame(ssm))
head(as.data.frame(ssm, long = FALSE))

# Type 2
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
```

```
    beta = beta,
    psi_l = psi_l,
    nu = nu,
    lambda = lambda,
    theta_l = theta_l,
    type = 2,
    x = x,
    gamma = gamma,
    kappa = kappa
  )

  head(as.data.frame(ssm))
  head(as.data.frame(ssm, long = FALSE))
```

---

as.matrix.simstatespace *Coerce an Object of Class* `simstatespace` *to a Matrix*

Description

Coerce an Object of Class `simstatespace` to a Matrix

Usage

```
## S3 method for class 'simstatespace'
as.matrix(x, eta = FALSE, long = TRUE, ...)
```

Arguments

`x`	Object of class `simstatespace`.
`eta`	Logical. If `eta = TRUE`, include `eta`. If `eta = FALSE`, exclude `eta`.
`long`	Logical. If `long = TRUE`, use long format. If `long = FALSE`, use wide format.
`...`	Additional arguments.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
# prepare parameters
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
## dynamic structure
```

```
p <- 3
mu0 <- rep(x = 0, times = p)
sigma0 <- diag(p)
sigma0_l <- t(chol(sigma0))
alpha <- rep(x = 0, times = p)
beta <- 0.50 * diag(p)
psi <- diag(p)
psi_l <- t(chol(psi))
## measurement model
k <- 3
nu <- rep(x = 0, times = k)
lambda <- diag(k)
theta <- 0.50 * diag(k)
theta_l <- t(chol(theta))
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
)
gamma <- diag(x = 0.10, nrow = p, ncol = j)
kappa <- diag(x = 0.10, nrow = k, ncol = j)

# Type 0
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)

head(as.matrix(ssm))
head(as.matrix(ssm, long = FALSE))

# Type 1
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
```

```
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
)

head(as.matrix(ssm))
head(as.matrix(ssm, long = FALSE))

# Type 2
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)

head(as.matrix(ssm))
head(as.matrix(ssm, long = FALSE))
```

`coef.statespacepb` *Estimated Parameter Method for an Object of Class* `statespacepb`

Description

Estimated Parameter Method for an Object of Class `statespacepb`

Usage

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

Arguments

`object`	Object of Class `statespacepb`.
`...`	additional arguments.

Value

Returns a vector of estimated parameters.

Author(s)

Ivan Jacob Agaloos Pesigan

`confint.statespacepb` *Confidence Intervals Method for an Object of Class* `statespacepb`

Description

Confidence Intervals Method for an Object of Class `statespacepb`

Usage

```
## S3 method for class 'statespacepb'
confint(object, parm = NULL, level = 0.95, type = "pc", ...)
```

Arguments

`object`	Object of Class `statespacepb`.
`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.
`type`	Charater string. Confidence interval type, that is, `type = "pc"` for percentile; `type = "bc"` for bias corrected.
`...`	additional 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.statespacepb` *Extract Method for an Object of Class* `statespacepb`

Description

Extract Method for an Object of Class `statespacepb`

Usage

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

Arguments

`object`	Object of Class `statespacepb`.
`what`	Character string. What specific matrix to extract. If `what = NULL`, extract all available matrices.

Value

Returns a list. Each element of the list is a list of bootstrap estimates in matrix format.

Author(s)

Ivan Jacob Agaloos Pesigan

`LinSDE2SSM` *Convert Parameters from the Linear Stochastic Differential Equation Model to State Space Model Parameterization*

Description

This function converts parameters from the linear stochastic differential equation model to state space model parameterization.

Usage

```
LinSDE2SSM(iota, phi, sigma_l, delta_t)
```

Arguments

`iota`	Numeric vector. An unobserved term that is constant over time ($\boldsymbol{\iota}$).
`phi`	Numeric matrix. The drift matrix which represents the rate of change of the solution in the absence of any random fluctuations ($\boldsymbol{\Phi}$).
`sigma_l`	Numeric matrix. Cholesky factorization (`t(chol(sigma))`) of the covariance matrix of volatility or randomness in the process ($\boldsymbol{\Sigma}$).
`delta_t`	Numeric. Time interval (Δ_t).

Details

Let the linear stochastic equation model be given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \left(\boldsymbol{\iota} + \boldsymbol{\Phi}\boldsymbol{\eta}_{i,t}\right)\mathrm{d}t + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}$$

for individual i and time t. The discrete-time state space model given below represents the discrete-time solution for the linear stochastic differential equation.

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

with

$$\boldsymbol{\beta}_{\Delta t_{l_i}} = \exp\left(\Delta t \boldsymbol{\Phi}\right),$$

$$\boldsymbol{\alpha}_{\Delta t_{l_i}} = \boldsymbol{\Phi}^{-1}\left(\boldsymbol{\beta} - \mathbf{I}_p\right)\boldsymbol{\iota}, \quad \text{and}$$

$$\mathrm{vec}\left(\boldsymbol{\Psi}_{\Delta t_{l_i}}\right) = \left[\left(\boldsymbol{\Phi} \otimes \mathbf{I}_p\right) + \left(\mathbf{I}_p \otimes \boldsymbol{\Phi}\right)\right]\left[\exp\left(\left[\left(\boldsymbol{\Phi} \otimes \mathbf{I}_p\right) + \left(\mathbf{I}_p \otimes \boldsymbol{\Phi}\right)\right]\Delta t\right) - \mathbf{I}_{p \times p}\right]\mathrm{vec}\left(\boldsymbol{\Sigma}\right)$$

where t denotes continuous-time processes that can be defined by any arbitrary time point, t_{l_i} the l^{th} observed measurement occassion for individual i, p the number of latent variables and Δt the time interval.

Value

Returns a list of state space parameters:

- `alpha`: Numeric vector. Vector of constant values for the dynamic model ($\boldsymbol{\alpha}$).
- `beta`: Numeric matrix. Transition matrix relating the values of the latent variables from the previous time point to the current time point. ($\boldsymbol{\beta}$).
- `psi_l`: Numeric matrix. Cholesky factorization (`t(chol(psi))`) of the process noise covariance matrix $\boldsymbol{\Psi}$.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Harvey, A. C. (1990). Forecasting, structural time series models and the Kalman filter. Cambridge University Press. doi:10.1017/cbo9781107049994

See Also

Other Simulation of State Space Models Data Functions: `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
p <- 2
iota <- c(0.317, 0.230)
phi <- matrix(
  data = c(
   -0.10,
   0.05,
   0.05,
   -0.10
 ),
 nrow = p
)
sigma <- matrix(
  data = c(
    2.79,
    0.06,
    0.06,
    3.27
  ),
  nrow = p
)
sigma_l <- t(chol(sigma))
delta_t <- 0.10
```

```
LinSDE2SSM(
  iota = iota,
  phi = phi,
  sigma_l = sigma_l,
  delta_t = delta_t
)
```

PBSSMFixed *Parametric Bootstrap for the State Space Model (Fixed Parameters)*

Description

This function simulates data from a state-space model and fits the model using the `dynr` package. The process is repeated `R` times. It assumes that the parameters remain constant across individuals and over time. At the moment, the function only supports `type = 0`.

Usage

```
PBSSMFixed(
  R,
  path,
  prefix,
  n,
  time,
  delta_t = 0.1,
  mu0,
  sigma0_l,
  alpha,
  beta,
  psi_l,
  nu,
  lambda,
  theta_l,
  type = 0,
  x = NULL,
  gamma = NULL,
  kappa = NULL,
  mu0_fixed = FALSE,
  sigma0_fixed = FALSE,
  alpha_level = 0.05,
  optimization_flag = TRUE,
  hessian_flag = FALSE,
  verbose = FALSE,
  weight_flag = FALSE,
  debug_flag = FALSE,
  perturb_flag = FALSE,
```

```
  xtol_rel = 1e-07,
  stopval = -9999,
  ftol_rel = -1,
  ftol_abs = -1,
  maxeval = as.integer(-1),
  maxtime = -1,
  ncores = NULL,
  seed = NULL
)
```

Arguments

`R`	Positive integer. Number of bootstrap samples.
`path`	Path to a directory to store bootstrap samples and estimates.
`prefix`	Character string. Prefix used for the file names for the bootstrap samples and estimates.
`n`	Positive integer. Number of individuals.
`time`	Positive integer. Number of time points.
`delta_t`	Numeric. Time interval. The default value is `1.0` with an option to use a numeric value for the discretized state space model parameterization of the linear stochastic differential equation model.
`mu0`	Numeric vector. Mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta}\|0}$).
`sigma0_l`	Numeric matrix. Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta}\|0}$).
`alpha`	Numeric vector. Vector of constant values for the dynamic model ($\boldsymbol{\alpha}$).
`beta`	Numeric matrix. Transition matrix relating the values of the latent variables at the previous to the current time point ($\boldsymbol{\beta}$).
`psi_l`	Numeric matrix. Cholesky factorization (`t(chol(psi))`) of the covariance matrix of the process noise ($\boldsymbol{\Psi}$).
`nu`	Numeric vector. Vector of intercept values for the measurement model ($\boldsymbol{\nu}$).
`lambda`	Numeric matrix. Factor loading matrix linking the latent variables to the observed variables ($\boldsymbol{\Lambda}$).
`theta_l`	Numeric matrix. Cholesky factorization (`t(chol(theta))`) of the covariance matrix of the measurement error ($\boldsymbol{\Theta}$).
`type`	Integer. State space model type. See Details for more information.
`x`	List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.
`gamma`	Numeric matrix. Matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
`kappa`	Numeric matrix. Matrix linking the covariates to the observed variables at current time point ($\boldsymbol{\kappa}$).
`mu0_fixed`	Logical. If `mu0_fixed = TRUE`, fix the initial mean vector to `mu0`. If `mu0_fixed = FALSE`, `mu0` is estimated.

`sigma0_fixed` Logical. If `sigma0_fixed = TRUE`, fix the initial covariance matrix to `tcrossprod(sigma0_l)`. If `sigma0_fixed = FALSE`, `sigma0` is estimated.

`alpha_level` Numeric vector. Significance level α.

`optimization_flag` a flag (TRUE/FALSE) indicating whether optimization is to be done.

`hessian_flag` a flag (TRUE/FALSE) indicating whether the Hessian matrix is to be calculated.

`verbose` a flag (TRUE/FALSE) indicating whether more detailed intermediate output during the estimation process should be printed

`weight_flag` a flag (TRUE/FALSE) indicating whether the negative log likelihood function should be weighted by the length of the time series for each individual

`debug_flag` a flag (TRUE/FALSE) indicating whether users want additional dynr output that can be used for diagnostic purposes

`perturb_flag` a flag (TRUE/FLASE) indicating whether to perturb the latent states during estimation. Only useful for ensemble forecasting.

`xtol_rel` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`stopval` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`ftol_rel` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`ftol_abs` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`maxeval` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`maxtime` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`ncores` Positive integer. Number of cores to use. If `ncores = NULL`, use a single core. Consider using multiple cores when number of bootstrap samples `R` is a large value.

`seed` Random seed.

Details

Type 0:

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. $\boldsymbol{\nu}$ denotes a vector of intercepts, $\boldsymbol{\Lambda}$ a matrix of factor loadings, and $\boldsymbol{\Theta}$ the covariance matrix of $\boldsymbol{\varepsilon}$.

An alternative representation of the measurement error is given by

$$\boldsymbol{\varepsilon}_{i,t} = \boldsymbol{\Theta}^{\frac{1}{2}} \mathbf{z}_{i,t}, \quad \text{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

where $\mathbf{z}_{i,t}$ is a vector of independent standard normal random variables and $\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)^{\prime} = \boldsymbol{\Theta}$.

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 $\boldsymbol{\zeta}_{i,t}$.

An alternative representation of the dynamic noise is given by

$$\boldsymbol{\zeta}_{i,t} = \boldsymbol{\Psi}^{\frac{1}{2}} \mathbf{z}_{i,t}, \quad \text{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

where $\left(\boldsymbol{\Psi}^{\frac{1}{2}}\right)\left(\boldsymbol{\Psi}^{\frac{1}{2}}\right)^{\prime} = \boldsymbol{\Psi}$.

Type 1:

The measurement model is given by

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

The dynamic structure is given by

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

where $\mathbf{x}_{i,t}$ represents a vector of covariates at time t and individual i, and $\boldsymbol{\Gamma}$ the coefficient matrix linking the covariates to the latent variables.

Type 2:

The measurement model is given by

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

where $\boldsymbol{\kappa}$ represents the coefficient matrix linking the covariates to the observed variables.

The dynamic structure is given by

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

Value

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

call Function call.

args Function arguments.

thetahatstar Sampling distribution of $\hat{\boldsymbol{\theta}}$.

vcov Sampling variance-covariance matrix of $\hat{\boldsymbol{\theta}}$.

est Vector of estimated $\hat{\boldsymbol{\theta}}$.

fun Function used ("PBSSMFixed").

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
## Not run:
# prepare parameters
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
## dynamic structure
p <- 3
mu0 <- rep(x = 0, times = p)
sigma0 <- 0.001 * diag(p)
sigma0_l <- t(chol(sigma0))
alpha <- rep(x = 0, times = p)
beta <- 0.50 * diag(p)
psi <- 0.001 * diag(p)
psi_l <- t(chol(psi))
## measurement model
k <- 3
nu <- rep(x = 0, times = k)
lambda <- diag(k)
theta <- 0.001 * diag(k)
theta_l <- t(chol(theta))

pb <- PBSSMFixed(
  R = 1000L,
  path = getwd(),
  prefix = "ssm",
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
```

```
    sigma0_l = sigma0_l,
    alpha = alpha,
    beta = beta,
    psi_l = psi_l,
    nu = nu,
    lambda = lambda,
    theta_l = theta_l,
    type = 0,
    ncores = parallel::detectCores() - 1,
    seed = 42
  )
  print(pb)
  summary(pb)
  confint(pb)
  vcov(pb)
  coef(pb)
  print(pb, type = "bc") # bias-corrected
  summary(pb, type = "bc")
  confint(pb, type = "bc")

  ## End(Not run)
```

`PBSSMLinSDEFixed` *Parametric Bootstrap for the Linear Stochastic Differential Equation Model using a State Space Model Parameterization (Fixed Parameters)*

Description

This function simulates data from a linear stochastic differential equation model using a state-space model parameterization and fits the model using the `dynr` package. The process is repeated `R` times. It assumes that the parameters remain constant across individuals and over time. At the moment, the function only supports `type = 0`.

Usage

```
PBSSMLinSDEFixed(
  R,
  path,
  prefix,
  n,
  time,
  delta_t = 0.1,
  mu0,
  sigma0_l,
  iota,
  phi,
  sigma_l,
```

```
  nu,
  lambda,
  theta_l,
  type = 0,
  x = NULL,
  gamma = NULL,
  kappa = NULL,
  mu0_fixed = FALSE,
  sigma0_fixed = FALSE,
  alpha_level = 0.05,
  optimization_flag = TRUE,
  hessian_flag = FALSE,
  verbose = FALSE,
  weight_flag = FALSE,
  debug_flag = FALSE,
  perturb_flag = FALSE,
  xtol_rel = 1e-07,
  stopval = -9999,
  ftol_rel = -1,
  ftol_abs = -1,
  maxeval = as.integer(-1),
  maxtime = -1,
  ncores = NULL,
  seed = NULL
)
```

Arguments

R	Positive integer. Number of bootstrap samples.
path	Path to a directory to store bootstrap samples and estimates.
prefix	Character string. Prefix used for the file names for the bootstrap samples and estimates.
n	Positive integer. Number of individuals.
time	Positive integer. Number of time points.
delta_t	Numeric. Time interval (Δ_t).
mu0	Numeric vector. Mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta}\mid 0}$).
sigma0_l	Numeric matrix. Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta}\mid 0}$).
iota	Numeric vector. An unobserved term that is constant over time ($\boldsymbol{\iota}$).
phi	Numeric matrix. The drift matrix which represents the rate of change of the solution in the absence of any random fluctuations ($\boldsymbol{\Phi}$).
sigma_l	Numeric matrix. Cholesky factorization (`t(chol(sigma))`) of the covariance matrix of volatility or randomness in the process ($\boldsymbol{\Sigma}$).
nu	Numeric vector. Vector of intercept values for the measurement model ($\boldsymbol{\nu}$).

lambda	Numeric matrix. Factor loading matrix linking the latent variables to the observed variables ($\boldsymbol{\Lambda}$).
theta_l	Numeric matrix. Cholesky factorization (`t(chol(theta))`) of the covariance matrix of the measurement error ($\boldsymbol{\Theta}$).
type	Integer. State space model type. See Details for more information.
x	List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.
gamma	Numeric matrix. Matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
kappa	Numeric matrix. Matrix linking the covariates to the observed variables at current time point ($\boldsymbol{\kappa}$).
mu0_fixed	Logical. If `mu0_fixed = TRUE`, fix the initial mean vector to `mu0`. If `mu0_fixed = FALSE`, `mu0` is estimated.
sigma0_fixed	Logical. If `sigma0_fixed = TRUE`, fix the initial covariance matrix to `tcrossprod(sigma0_l)`. If `sigma0_fixed = FALSE`, `sigma0` is estimated.
alpha_level	Numeric vector. Significance level α.
optimization_flag	a flag (TRUE/FALSE) indicating whether optimization is to be done.
hessian_flag	a flag (TRUE/FALSE) indicating whether the Hessian matrix is to be calculated.
verbose	a flag (TRUE/FALSE) indicating whether more detailed intermediate output during the estimation process should be printed
weight_flag	a flag (TRUE/FALSE) indicating whether the negative log likelihood function should be weighted by the length of the time series for each individual
debug_flag	a flag (TRUE/FALSE) indicating whether users want additional dynr output that can be used for diagnostic purposes
perturb_flag	a flag (TRUE/FLASE) indicating whether to perturb the latent states during estimation. Only useful for ensemble forecasting.
xtol_rel	Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.
stopval	Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.
ftol_rel	Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.
ftol_abs	Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.
maxeval	Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.
maxtime	Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.
ncores	Positive integer. Number of cores to use. If `ncores = NULL`, use a single core. Consider using multiple cores when number of bootstrap samples `R` is a large value.
seed	Random seed.

Details

Type 0:

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. $\boldsymbol{\nu}$ denotes a vector of intercepts, $\boldsymbol{\Lambda}$ a matrix of factor loadings, and $\boldsymbol{\Theta}$ the covariance matrix of $\boldsymbol{\varepsilon}$.

An alternative representation of the measurement error is given by

$$\boldsymbol{\varepsilon}_{i,t} = \boldsymbol{\Theta}^{\frac{1}{2}}\mathbf{z}_{i,t}, \quad \text{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

where $\mathbf{z}_{i,t}$ is a vector of independent standard normal random variables and $\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)' = \boldsymbol{\Theta}$.

The dynamic structure is given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \left(\boldsymbol{\iota} + \boldsymbol{\Phi}\boldsymbol{\eta}_{i,t}\right)\mathrm{d}t + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}$$

where $\boldsymbol{\iota}$ is a term which is unobserved and constant over time, $\boldsymbol{\Phi}$ is the drift matrix which represents the rate of change of the solution in the absence of any random fluctuations, $\boldsymbol{\Sigma}$ is the matrix of volatility or randomness in the process, and $\mathrm{d}\boldsymbol{W}$ is a Wiener process or Brownian motion, which represents random fluctuations.

Type 1:

The measurement model is given by

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

The dynamic structure is given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \left(\boldsymbol{\iota} + \boldsymbol{\Phi}\boldsymbol{\eta}_{i,t}\right)\mathrm{d}t + \boldsymbol{\Gamma}\mathbf{x}_{i,t} + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}$$

where $\mathbf{x}_{i,t}$ represents a vector of covariates at time t and individual i, and $\boldsymbol{\Gamma}$ the coefficient matrix linking the covariates to the latent variables.

Type 2:

The measurement model is given by

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

where $\boldsymbol{\kappa}$ represents the coefficient matrix linking the covariates to the observed variables.

The dynamic structure is given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \left(\boldsymbol{\iota} + \boldsymbol{\Phi}\boldsymbol{\eta}_{i,t}\right)\mathrm{d}t + \boldsymbol{\Gamma}\mathbf{x}_{i,t} + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}.$$

State Space Parameterization:
The state space parameters as a function of the linear stochastic differential equation model parameters are given by

$$\boldsymbol{\beta}_{\Delta t_{l_i}} = \exp\left(\Delta t \boldsymbol{\Phi}\right)$$

$$\boldsymbol{\alpha}_{\Delta t_{l_i}} = \boldsymbol{\Phi}^{-1}\left(\boldsymbol{\beta} - \mathbf{I}_p\right)\boldsymbol{\iota}$$

$$\mathrm{vec}\left(\boldsymbol{\Psi}_{\Delta t_{l_i}}\right) = \left[\left(\boldsymbol{\Phi} \otimes \mathbf{I}_p\right) + \left(\mathbf{I}_p \otimes \boldsymbol{\Phi}\right)\right]\left[\exp\left(\left[\left(\boldsymbol{\Phi} \otimes \mathbf{I}_p\right) + \left(\mathbf{I}_p \otimes \boldsymbol{\Phi}\right)\right]\Delta t\right) - \mathbf{I}_{p \times p}\right]\mathrm{vec}\left(\boldsymbol{\Sigma}\right)$$

where p is the number of latent variables and Δt is the time interval.

Value

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

call Function call.

args Function arguments.

thetahatstar Sampling distribution of $\hat{\boldsymbol{\theta}}$.

vcov Sampling variance-covariance matrix of $\hat{\boldsymbol{\theta}}$.

est Vector of estimated $\hat{\boldsymbol{\theta}}$.

fun Function used ("PBSSMLinSDEFixed").

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

Chow, S.-M., Losardo, D., Park, J., & Molenaar, P. C. M. (2023). Continuous-time dynamic models: Connections to structural equation models and other discrete-time models. In R. H. Hoyle (Ed.), Handbook of structural equation modeling (2nd ed.). The Guilford Press.

Harvey, A. C. (1990). Forecasting, structural time series models and the Kalman filter. Cambridge University Press. doi:10.1017/cbo9781107049994

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
## Not run:
# prepare parameters
## number of individuals
n <- 5
## time points
time <- 50
delta_t <- 0.10
## dynamic structure
p <- 2
mu0 <- c(-3.0, 1.5)
sigma0 <- 0.001 * diag(p)
sigma0_l <- t(chol(sigma0))
iota <- c(0.317, 0.230)
phi <- matrix(
  data = c(
    -0.10,
    0.05,
    0.05,
    -0.10
  ),
  nrow = p
)
sigma <- matrix(
  data = c(
    2.79,
    0.06,
    0.06,
    3.27
  ),
  nrow = p
)
sigma_l <- t(chol(sigma))
## measurement model
k <- 2
nu <- rep(x = 0, times = k)
lambda <- diag(k)
theta <- 0.001 * diag(k)
theta_l <- t(chol(theta))

pb <- PBSSMLinSDEFixed(
  R = 1000L,
  path = getwd(),
  prefix = "lse",
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  iota = iota,
  phi = phi,
  sigma_l = sigma_l,
```

```
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0,
  ncores = parallel::detectCores() - 1,
  seed = 42
)
print(pb)
summary(pb)
confint(pb)
vcov(pb)
coef(pb)
print(pb, type = "bc") # bias-corrected
summary(pb, type = "bc")
confint(pb, type = "bc")

## End(Not run)
```

`PBSSMOUFixed` *Parametric Bootstrap for the Ornstein–Uhlenbeck Model using a State Space Model Parameterization (Fixed Parameters)*

Description

This function simulates data from a Ornstein–Uhlenbeck (OU) model using a state-space model parameterization and fits the model using the `dynr` package. The process is repeated `R` times. It assumes that the parameters remain constant across individuals and over time. At the moment, the function only supports `type = 0`.

Usage

```
PBSSMOUFixed(
  R,
  path,
  prefix,
  n,
  time,
  delta_t = 0.1,
  mu0,
  sigma0_l,
  mu,
  phi,
  sigma_l,
  nu,
  lambda,
  theta_l,
  type = 0,
```

```
  x = NULL,
  gamma = NULL,
  kappa = NULL,
  mu0_fixed = FALSE,
  sigma0_fixed = FALSE,
  alpha_level = 0.05,
  optimization_flag = TRUE,
  hessian_flag = FALSE,
  verbose = FALSE,
  weight_flag = FALSE,
  debug_flag = FALSE,
  perturb_flag = FALSE,
  xtol_rel = 1e-07,
  stopval = -9999,
  ftol_rel = -1,
  ftol_abs = -1,
  maxeval = as.integer(-1),
  maxtime = -1,
  ncores = NULL,
  seed = NULL
)
```

Arguments

`R`	Positive integer. Number of bootstrap samples.
`path`	Path to a directory to store bootstrap samples and estimates.
`prefix`	Character string. Prefix used for the file names for the bootstrap samples and estimates.
`n`	Positive integer. Number of individuals.
`time`	Positive integer. Number of time points.
`delta_t`	Numeric. Time interval (Δ_t).
`mu0`	Numeric vector. Mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta}\|0}$).
`sigma0_l`	Numeric matrix. Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta}\|0}$).
`mu`	Numeric vector. The long-term mean or equilibrium level ($\boldsymbol{\mu}$).
`phi`	Numeric matrix. The drift matrix which represents the rate of change of the solution in the absence of any random fluctuations ($\boldsymbol{\Phi}$). It also represents the rate of mean reversion, determining how quickly the variable returns to its mean.
`sigma_l`	Numeric matrix. Cholesky factorization (`t(chol(sigma))`) of the covariance matrix of volatility or randomness in the process ($\boldsymbol{\Sigma}$).
`nu`	Numeric vector. Vector of intercept values for the measurement model ($\boldsymbol{\nu}$).
`lambda`	Numeric matrix. Factor loading matrix linking the latent variables to the observed variables ($\boldsymbol{\Lambda}$).

theta_l	Numeric matrix. Cholesky factorization (t(chol(theta))) of the covariance matrix of the measurement error ($\boldsymbol{\Theta}$).
type	Integer. State space model type. See Details for more information.
x	List. Each element of the list is a matrix of covariates for each individual i in n. The number of columns in each matrix should be equal to time.
gamma	Numeric matrix. Matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
kappa	Numeric matrix. Matrix linking the covariates to the observed variables at current time point ($\boldsymbol{\kappa}$).
mu0_fixed	Logical. If mu0_fixed = TRUE, fix the initial mean vector to mu0. If mu0_fixed = FALSE, mu0 is estimated.
sigma0_fixed	Logical. If sigma0_fixed = TRUE, fix the initial covariance matrix to tcrossprod(sigma0_l). If sigma0_fixed = FALSE, sigma0 is estimated.
alpha_level	Numeric vector. Significance level α.
optimization_flag	a flag (TRUE/FALSE) indicating whether optimization is to be done.
hessian_flag	a flag (TRUE/FALSE) indicating whether the Hessian matrix is to be calculated.
verbose	a flag (TRUE/FALSE) indicating whether more detailed intermediate output during the estimation process should be printed
weight_flag	a flag (TRUE/FALSE) indicating whether the negative log likelihood function should be weighted by the length of the time series for each individual
debug_flag	a flag (TRUE/FALSE) indicating whether users want additional dynr output that can be used for diagnostic purposes
perturb_flag	a flag (TRUE/FLASE) indicating whether to perturb the latent states during estimation. Only useful for ensemble forecasting.
xtol_rel	Stopping criteria option for parameter optimization. See dynr::dynr.model() for more details.
stopval	Stopping criteria option for parameter optimization. See dynr::dynr.model() for more details.
ftol_rel	Stopping criteria option for parameter optimization. See dynr::dynr.model() for more details.
ftol_abs	Stopping criteria option for parameter optimization. See dynr::dynr.model() for more details.
maxeval	Stopping criteria option for parameter optimization. See dynr::dynr.model() for more details.
maxtime	Stopping criteria option for parameter optimization. See dynr::dynr.model() for more details.
ncores	Positive integer. Number of cores to use. If ncores = NULL, use a single core. Consider using multiple cores when number of bootstrap samples R is a large value.
seed	Random seed.

Details

Type 0:

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. $\boldsymbol{\nu}$ denotes a vector of intercepts, $\boldsymbol{\Lambda}$ a matrix of factor loadings, and $\boldsymbol{\Theta}$ the covariance matrix of $\boldsymbol{\varepsilon}$.

An alternative representation of the measurement error is given by

$$\boldsymbol{\varepsilon}_{i,t} = \boldsymbol{\Theta}^{\frac{1}{2}}\mathbf{z}_{i,t}, \quad \text{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

where $\mathbf{z}_{i,t}$ is a vector of independent standard normal random variables and $\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)' = \boldsymbol{\Theta}$.

The dynamic structure is given by

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

where $\boldsymbol{\mu}$ is the long-term mean or equilibrium level, $\boldsymbol{\Phi}$ is the rate of mean reversion, determining how quickly the variable returns to its mean, $\boldsymbol{\Sigma}$ is the matrix of volatility or randomness in the process, and $\mathrm{d}\boldsymbol{W}$ is a Wiener process or Brownian motion, which represents random fluctuations.

Type 1:

The measurement model is given by

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

The dynamic structure is given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \boldsymbol{\Phi}\left(\boldsymbol{\eta}_{i,t} - \boldsymbol{\mu}\right)\mathrm{d}t + \boldsymbol{\Gamma}\mathbf{x}_{i,t} + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}$$

where $\mathbf{x}_{i,t}$ represents a vector of covariates at time t and individual i, and $\boldsymbol{\Gamma}$ the coefficient matrix linking the covariates to the latent variables.

Type 2:

The measurement model is given by

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

where $\boldsymbol{\kappa}$ represents the coefficient matrix linking the covariates to the observed variables.

The dynamic structure is given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \boldsymbol{\Phi}\left(\boldsymbol{\eta}_{i,t} - \boldsymbol{\mu}\right)\mathrm{d}t + \boldsymbol{\Gamma}\mathbf{x}_{i,t} + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}.$$

The OU model as a linear stochastic differential equation model:

The OU model is a first-order linear stochastic differential equation model in the form of

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \left(\boldsymbol{\iota} + \boldsymbol{\Phi}\boldsymbol{\eta}_{i,t}\right)\mathrm{d}t + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}$$

where $\boldsymbol{\mu} = -\boldsymbol{\Phi}^{-1}\boldsymbol{\iota}$ and, equivalently $\boldsymbol{\iota} = -\boldsymbol{\Phi}\boldsymbol{\mu}$.

Value

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

call Function call.

args Function arguments.

thetahatstar Sampling distribution of $\hat{\boldsymbol{\theta}}$.

vcov Sampling variance-covariance matrix of $\hat{\boldsymbol{\theta}}$.

est Vector of estimated $\hat{\boldsymbol{\theta}}$.

fun Function used ("PBSSMOUFixed").

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

Chow, S.-M., Losardo, D., Park, J., & Molenaar, P. C. M. (2023). Continuous-time dynamic models: Connections to structural equation models and other discrete-time models. In R. H. Hoyle (Ed.), Handbook of structural equation modeling (2nd ed.). The Guilford Press.

Harvey, A. C. (1990). Forecasting, structural time series models and the Kalman filter. Cambridge University Press. doi:10.1017/cbo9781107049994

Oravecz, Z., Tuerlinckx, F., & Vandekerckhove, J. (2011). A hierarchical latent stochastic differential equation model for affective dynamics. Psychological Methods, 16 (4), 468–490. doi:10.1037/a0024375

Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the theory of the brownian motion. Physical Review, 36 (5), 823–841. doi:10.1103/physrev.36.823

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
## Not run:
# prepare parameters
## number of individuals
n <- 5
## time points
time <- 50
```

```
delta_t <- 0.10
## dynamic structure
p <- 2
mu0 <- c(-3.0, 1.5)
sigma0 <- 0.001 * diag(p)
sigma0_l <- t(chol(sigma0))
mu <- c(5.76, 5.18)
phi <- matrix(
  data = c(
    -0.10,
    0.05,
    0.05,
    -0.10
  ),
  nrow = p
)
sigma <- matrix(
  data = c(
    2.79,
    0.06,
    0.06,
    3.27
  ),
  nrow = p
)
sigma_l <- t(chol(sigma))
## measurement model
k <- 2
nu <- rep(x = 0, times = k)
lambda <- diag(k)
theta <- 0.001 * diag(k)
theta_l <- t(chol(theta))

pb <- PBSSMOUFixed(
  R = 1000L,
  path = getwd(),
  prefix = "ou",
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  mu = mu,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0,
  ncores = parallel::detectCores() - 1,
  seed = 42
)
print(pb)
```

```
summary(pb)
confint(pb)
vcov(pb)
coef(pb)
print(pb, type = "bc") # bias-corrected
summary(pb, type = "bc")
confint(pb, type = "bc")

## End(Not run)
```

PBSSMVARFixed *Parametric Bootstrap for the Vector Autoregressive Model (Fixed Parameters)*

Description

This function simulates data from a vector autoregressive model using a state-space model parameterization and fits the model using the `dynr` package. The process is repeated `R` times. It assumes that the parameters remain constant across individuals and over time. At the moment, the function only supports `type = 0`.

Usage

```
PBSSMVARFixed(
  R,
  path,
  prefix,
  n,
  time,
  mu0,
  sigma0_l,
  alpha,
  beta,
  psi_l,
  type = 0,
  x = NULL,
  gamma = NULL,
  mu0_fixed = FALSE,
  sigma0_fixed = FALSE,
  alpha_level = 0.05,
  optimization_flag = TRUE,
  hessian_flag = FALSE,
  verbose = FALSE,
  weight_flag = FALSE,
  debug_flag = FALSE,
  perturb_flag = FALSE,
  xtol_rel = 1e-07,
```

```
  stopval = -9999,
  ftol_rel = -1,
  ftol_abs = -1,
  maxeval = as.integer(-1),
  maxtime = -1,
  ncores = NULL,
  seed = NULL
)
```

Arguments

R	Positive integer. Number of bootstrap samples.
path	Path to a directory to store bootstrap samples and estimates.
prefix	Character string. Prefix used for the file names for the bootstrap samples and estimates.
n	Positive integer. Number of individuals.
time	Positive integer. Number of time points.
mu0	Numeric vector. Mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta}\mid 0}$).
sigma0_l	Numeric matrix. Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta}\mid 0}$).
alpha	Numeric vector. Vector of constant values for the dynamic model ($\boldsymbol{\alpha}$).
beta	Numeric matrix. Transition matrix relating the values of the latent variables at the previous to the current time point ($\boldsymbol{\beta}$).
psi_l	Numeric matrix. Cholesky factorization (`t(chol(psi))`) of the covariance matrix of the process noise ($\boldsymbol{\Psi}$).
type	Integer. State space model type. See Details for more information.
x	List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.
gamma	Numeric matrix. Matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
mu0_fixed	Logical. If `mu0_fixed = TRUE`, fix the initial mean vector to `mu0`. If `mu0_fixed = FALSE`, `mu0` is estimated.
sigma0_fixed	Logical. If `sigma0_fixed = TRUE`, fix the initial covariance matrix to `tcrossprod(sigma0_l)`. If `sigma0_fixed = FALSE`, `sigma0` is estimated.
alpha_level	Numeric vector. Significance level α.
optimization_flag	a flag (TRUE/FALSE) indicating whether optimization is to be done.
hessian_flag	a flag (TRUE/FALSE) indicating whether the Hessian matrix is to be calculated.
verbose	a flag (TRUE/FALSE) indicating whether more detailed intermediate output during the estimation process should be printed
weight_flag	a flag (TRUE/FALSE) indicating whether the negative log likelihood function should be weighted by the length of the time series for each individual

`debug_flag` a flag (TRUE/FALSE) indicating whether users want additional dynr output that can be used for diagnostic purposes

`perturb_flag` a flag (TRUE/FLASE) indicating whether to perturb the latent states during estimation. Only useful for ensemble forecasting.

`xtol_rel` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`stopval` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`ftol_rel` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`ftol_abs` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`maxeval` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`maxtime` Stopping criteria option for parameter optimization. See `dynr::dynr.model()` for more details.

`ncores` Positive integer. Number of cores to use. If `ncores = NULL`, use a single core. Consider using multiple cores when number of bootstrap samples `R` is a large value.

`seed` Random seed.

Details

Type 0:

The measurement model is given by

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

where $\mathbf{y}_{i,t}$ represents a vector of observed variables and $\boldsymbol{\eta}_{i,t}$ a vector of latent variables for individual i and time t. Since the observed and latent variables are equal, we only generate data from the dynamic structure.

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 $\boldsymbol{\zeta}_{i,t}$.

An alternative representation of the dynamic noise is given by

$$\boldsymbol{\zeta}_{i,t} = \boldsymbol{\Psi}^{\frac{1}{2}}\mathbf{z}_{i,t}, \quad \text{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

where $\left(\boldsymbol{\Psi}^{\frac{1}{2}}\right)\left(\boldsymbol{\Psi}^{\frac{1}{2}}\right)' = \boldsymbol{\Psi}$.

Type 1:
The measurement model is given by

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

The dynamic structure is given by

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

where $\mathbf{x}_{i,t}$ represents a vector of covariates at time t and individual i, and $\boldsymbol{\Gamma}$ the coefficient matrix linking the covariates to the latent variables.

Value

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

call Function call.

args Function arguments.

thetahatstar Sampling distribution of $\hat{\boldsymbol{\theta}}$.

vcov Sampling variance-covariance matrix of $\hat{\boldsymbol{\theta}}$.

est Vector of estimated $\hat{\boldsymbol{\theta}}$.

fun Function used ("PBSSMVARFixed").

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
## Not run:
# prepare parameters
## number of individuals
n <- 5
## time points
time <- 50
```

```
## dynamic structure
p <- 3
mu0 <- rep(x = 0, times = p)
sigma0 <- 0.001 * diag(p)
sigma0_l <- t(chol(sigma0))
alpha <- rep(x = 0, times = p)
beta <- 0.50 * diag(p)
psi <- 0.001 * diag(p)
psi_l <- t(chol(psi))

boot <- PBSSMVARFixed(
  R = 1000L,
  path = getwd(),
  prefix = "var",
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  type = 0,
  ncores = parallel::detectCores() - 1,
  seed = 42
)
print(pb)
summary(pb)
confint(pb)
vcov(pb)
coef(pb)
print(pb, type = "bc") # bias-corrected
summary(pb, type = "bc")
confint(pb, type = "bc")

## End(Not run)
```

---

plot.simstatespace *Plot Method for an Object of Class* simstatespace

---

Description

Plot Method for an Object of Class simstatespace

Usage

```
## S3 method for class 'simstatespace'
plot(x, id = NULL, time = NULL, eta = FALSE, type = "b", ...)
```

Arguments

x	Object of class `simstatespace`.
id	Numeric vector. Optional `id` numbers to plot. If `id = NULL`, plot all available data.
time	Numeric vector. Optional `time` points to plot. If `time = NULL`, plot all available data.
eta	Logical. If `eta = TRUE`, plot the latent variables. If `eta = FALSE`, plot the observed variables.
type	Character indicating the type of plotting; actually any of the types as in `plot.default()`.
...	Additional arguments.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
# prepare parameters
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
## dynamic structure
p <- 3
mu0 <- rep(x = 0, times = p)
sigma0 <- diag(p)
sigma0_l <- t(chol(sigma0))
alpha <- rep(x = 0, times = p)
beta <- 0.50 * diag(p)
psi <- diag(p)
psi_l <- t(chol(psi))
## measurement model
k <- 3
nu <- rep(x = 0, times = k)
lambda <- diag(k)
theta <- 0.50 * diag(k)
theta_l <- t(chol(theta))
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
```

```
)
gamma <- diag(x = 0.10, nrow = p, ncol = j)
kappa <- diag(x = 0.10, nrow = k, ncol = j)

# Type 0
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)

plot(ssm)
plot(ssm, id = 1:3, time = 0:9)

# Type 1
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
)

plot(ssm)
plot(ssm, id = 1:3, time = 0:9)

# Type 2
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
```

```
  theta_l = theta_l,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)

plot(ssm)
plot(ssm, id = 1:3, time = 0:9)
```

print.simstatespace *Print Method for an Object of Class* simstatespace

Description

Print Method for an Object of Class `simstatespace`

Usage

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

Arguments

`x`	Object of Class `simstatespace`.
`...`	Additional arguments.

Value

Prints simulated data in long format.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
# prepare parameters
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
## dynamic structure
p <- 3
mu0 <- rep(x = 0, times = p)
sigma0 <- diag(p)
sigma0_l <- t(chol(sigma0))
alpha <- rep(x = 0, times = p)
```

```
beta <- 0.50 * diag(p)
psi <- diag(p)
psi_l <- t(chol(psi))
## measurement model
k <- 3
nu <- rep(x = 0, times = k)
lambda <- diag(k)
theta <- 0.50 * diag(k)
theta_l <- t(chol(theta))
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
)
gamma <- diag(x = 0.10, nrow = p, ncol = j)
kappa <- diag(x = 0.10, nrow = k, ncol = j)

# Type 0
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)

print(ssm)

# Type 1
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
```

```
  type = 1,
  x = x,
  gamma = gamma
)

print(ssm)

# Type 2
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)

print(ssm)
```

`print.statespacepb` *Print Method for an Object of Class* `statespacepb`

Description

Print Method for an Object of Class `statespacepb`

Usage

```
## S3 method for class 'statespacepb'
print(x, alpha = NULL, type = "pc", digits = 4, ...)
```

Arguments

`x`	Object of Class `statespacepb`.
`alpha`	Numeric vector. Significance level α. If `alpha = NULL`, use the argument `alpha` used in `x`.
`type`	Charater string. Confidence interval type, that is, `type = "pc"` for percentile; `type = "bc"` for bias corrected.
`digits`	Digits to print.
`...`	additional arguments.

Value

Prints a matrix of estimates, standard errors, number of bootstrap replications, and confidence intervals.

Author(s)

Ivan Jacob Agaloos Pesigan

SimBetaN *Simulate Transition Matrices from the Multivariate Normal Distribution*

Description

This function simulates random transition matrices from the multivariate normal distribution. The function ensures that the generated transition matrices are stationary using `TestStationarity()`.

Usage

```
SimBetaN(n, beta, vcov_beta_vec_l)
```

Arguments

`n`	Positive integer. Number of replications.
`beta`	Numeric matrix. The transition matrix ($\boldsymbol{\beta}$).
`vcov_beta_vec_l`	Numeric matrix. Cholesky factorization (`t(chol(vcov_beta_vec))`) of the sampling variance-covariance matrix $\mathrm{vec}\left(\boldsymbol{\beta}\right)$.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
n <- 10
beta <- matrix(
  data = c(
    0.7, 0.5, -0.1,
    0.0, 0.6, 0.4,
    0, 0, 0.5
  ),
  nrow = 3
)
vcov_beta_vec_l <- t(chol(0.001 * diag(9)))
SimBetaN(n = n, beta = beta, vcov_beta_vec_l = vcov_beta_vec_l)
```

SimPhiN — *Simulate Random Drift Matrices from the Multivariate Normal Distribution*

Description

This function simulates random drift matrices from the multivariate normal distribution. The function ensures that the generated drift matrices are stable using `TestPhi()`.

Usage

```
SimPhiN(n, phi, vcov_phi_vec_l)
```

Arguments

`n`	Positive integer. Number of replications.
`phi`	Numeric matrix. The drift matrix ($\boldsymbol{\Phi}$).
`vcov_phi_vec_l`	Numeric matrix. Cholesky factorization (`t(chol(vcov_phi_vec))`) of the sampling variance-covariance matrix $\mathrm{vec}\left(\boldsymbol{\Phi}\right)$.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
n <- 10
phi <- matrix(
  data = c(
    -0.357, 0.771, -0.450,
    0.0, -0.511, 0.729,
    0, 0, -0.693
  ),
  nrow = 3
)
vcov_phi_vec_l <- t(chol(0.001 * diag(9)))
SimPhiN(n = n, phi = phi, vcov_phi_vec_l = vcov_phi_vec_l)
```

`SimSSMFixed` *Simulate Data from a State Space Model (Fixed Parameters)*

Description

This function simulates data using a state space model. It assumes that the parameters remain constant across individuals and over time.

Usage

```
SimSSMFixed(
  n,
  time,
  delta_t = 1,
  mu0,
  sigma0_l,
  alpha,
  beta,
  psi_l,
  nu,
  lambda,
  theta_l,
  type = 0,
  x = NULL,
  gamma = NULL,
  kappa = NULL
)
```

Arguments

`n`	Positive integer. Number of individuals.
`time`	Positive integer. Number of time points.

delta_t	Numeric. Time interval. The default value is `1.0` with an option to use a numeric value for the discretized state space model parameterization of the linear stochastic differential equation model.
mu0	Numeric vector. Mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta}\mid 0}$).
sigma0_l	Numeric matrix. Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta}\mid 0}$).
alpha	Numeric vector. Vector of constant values for the dynamic model ($\boldsymbol{\alpha}$).
beta	Numeric matrix. Transition matrix relating the values of the latent variables at the previous to the current time point ($\boldsymbol{\beta}$).
psi_l	Numeric matrix. Cholesky factorization (`t(chol(psi))`) of the covariance matrix of the process noise ($\boldsymbol{\Psi}$).
nu	Numeric vector. Vector of intercept values for the measurement model ($\boldsymbol{\nu}$).
lambda	Numeric matrix. Factor loading matrix linking the latent variables to the observed variables ($\boldsymbol{\Lambda}$).
theta_l	Numeric matrix. Cholesky factorization (`t(chol(theta))`) of the covariance matrix of the measurement error ($\boldsymbol{\Theta}$).
type	Integer. State space model type. See Details for more information.
x	List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.
gamma	Numeric matrix. Matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
kappa	Numeric matrix. Matrix linking the covariates to the observed variables at current time point ($\boldsymbol{\kappa}$).

Details

Type 0:

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. $\boldsymbol{\nu}$ denotes a vector of intercepts, $\boldsymbol{\Lambda}$ a matrix of factor loadings, and $\boldsymbol{\Theta}$ the covariance matrix of $\boldsymbol{\varepsilon}$.

An alternative representation of the measurement error is given by

$$\boldsymbol{\varepsilon}_{i,t} = \boldsymbol{\Theta}^{\frac{1}{2}}\mathbf{z}_{i,t}, \quad \text{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

where $\mathbf{z}_{i,t}$ is a vector of independent standard normal random variables and $\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)^{\prime} = \boldsymbol{\Theta}$.

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 $\boldsymbol{\zeta}_{i,t}$.

An alternative representation of the dynamic noise is given by

$$\boldsymbol{\zeta}_{i,t} = \boldsymbol{\Psi}^{\frac{1}{2}} \mathbf{z}_{i,t}, \quad \text{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

where $\left(\boldsymbol{\Psi}^{\frac{1}{2}}\right)\left(\boldsymbol{\Psi}^{\frac{1}{2}}\right)' = \boldsymbol{\Psi}$.

Type 1:

The measurement model is given by

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

The dynamic structure is given by

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

where $\mathbf{x}_{i,t}$ represents a vector of covariates at time t and individual i, and $\boldsymbol{\Gamma}$ the coefficient matrix linking the covariates to the latent variables.

Type 2:

The measurement model is given by

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

where $\boldsymbol{\kappa}$ represents the coefficient matrix linking the covariates to the observed variables.

The dynamic structure is given by

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

Value

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

- `call`: Function call.
- `args`: Function arguments.
- `data`: Generated data which is a list of length `n`. Each element of `data` is a list with the following elements:
  - `id`: A vector of ID numbers with length `l`, where `l` is the value of the function argument `time`.
  - `time`: A vector time points of length `l`.
  - `y`: A `l` by `k` matrix of values for the manifest variables.
  - `eta`: A `l` by `p` matrix of values for the latent variables.
  - `x`: A `l` by `j` matrix of values for the covariates (when covariates are included).
- `fun`: Function used.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
# prepare parameters
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
## dynamic structure
p <- 3
mu0 <- rep(x = 0, times = p)
sigma0 <- 0.001 * diag(p)
sigma0_l <- t(chol(sigma0))
alpha <- rep(x = 0, times = p)
beta <- 0.50 * diag(p)
psi <- 0.001 * diag(p)
psi_l <- t(chol(psi))
## measurement model
k <- 3
nu <- rep(x = 0, times = k)
lambda <- diag(k)
theta <- 0.001 * diag(k)
theta_l <- t(chol(theta))
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
```

```
  }
)
gamma <- diag(x = 0.10, nrow = p, ncol = j)
kappa <- diag(x = 0.10, nrow = k, ncol = j)

# Type 0
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)

plot(ssm)

# Type 1
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
)

plot(ssm)

# Type 2
ssm <- SimSSMFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
```

```
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)

plot(ssm)
```

`SimSSMIVary` *Simulate Data from a State Space Model (Individual-Varying Parameters)*

Description

This function simulates data using a state space model. It assumes that the parameters can vary across individuals.

Usage

```
SimSSMIVary(
  n,
  time,
  delta_t = 1,
  mu0,
  sigma0_l,
  alpha,
  beta,
  psi_l,
  nu,
  lambda,
  theta_l,
  type = 0,
  x = NULL,
  gamma = NULL,
  kappa = NULL
)
```

Arguments

`n`	Positive integer. Number of individuals.
`time`	Positive integer. Number of time points.
`delta_t`	Numeric. Time interval. The default value is `1.0` with an option to use a numeric value for the discretized state space model parameterization of the linear stochastic differential equation model.
`mu0`	List of numeric vectors. Each element of the list is the mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta}\|0}$).

sigma0_l	List of numeric matrices. Each element of the list is the Cholesky factorization (t(chol(sigma0))) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta}\mid 0}$).
alpha	List of numeric vectors. Each element of the list is the vector of constant values for the dynamic model ($\boldsymbol{\alpha}$).
beta	List of numeric matrices. Each element of the list is the transition matrix relating the values of the latent variables at the previous to the current time point ($\boldsymbol{\beta}$).
psi_l	List of numeric matrices. Each element of the list is the Cholesky factorization (t(chol(psi))) of the covariance matrix of the process noise ($\boldsymbol{\Psi}$).
nu	List of numeric vectors. Each element of the list is the vector of intercept values for the measurement model ($\boldsymbol{\nu}$).
lambda	List of numeric matrices. Each element of the list is the factor loading matrix linking the latent variables to the observed variables ($\boldsymbol{\Lambda}$).
theta_l	List of numeric matrices. Each element of the list is the Cholesky factorization (t(chol(theta))) of the covariance matrix of the measurement error ($\boldsymbol{\Theta}$).
type	Integer. State space model type. See Details in SimSSMFixed() for more information.
x	List. Each element of the list is a matrix of covariates for each individual i in n. The number of columns in each matrix should be equal to time.
gamma	List of numeric matrices. Each element of the list is the matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
kappa	List of numeric matrices. Each element of the list is the matrix linking the covariates to the observed variables at current time point ($\boldsymbol{\kappa}$).

Details

Parameters can vary across individuals by providing a list of parameter values. If the length of any of the parameters (mu0, sigma0_l, alpha, beta, psi_l, nu, lambda, theta_l, gamma, or kappa) is less the n, the function will cycle through the available values.

Value

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

- call: Function call.
- args: Function arguments.
- data: Generated data which is a list of length n. Each element of data is a list with the following elements:
  - id: A vector of ID numbers with length l, where l is the value of the function argument time.
  - time: A vector time points of length l.
  - y: A l by k matrix of values for the manifest variables.

  - `eta`: A `l` by `p` matrix of values for the latent variables.
  - `x`: A `l` by `j` matrix of values for the covariates (when covariates are included).
- `fun`: Function used.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
# prepare parameters
# In this example, beta varies across individuals.
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
## dynamic structure
p <- 3
mu0 <- list(
  rep(x = 0, times = p)
)
sigma0 <- 0.001 * diag(p)
sigma0_l <- list(
  t(chol(sigma0))
)
alpha <- list(
  rep(x = 0, times = p)
)
beta <- list(
  0.1 * diag(p),
  0.2 * diag(p),
  0.3 * diag(p),
  0.4 * diag(p),
  0.5 * diag(p)
)
```

```
psi <- 0.001 * diag(p)
psi_l <- list(
  t(chol(psi))
)
## measurement model
k <- 3
nu <- list(
  rep(x = 0, times = k)
)
lambda <- list(
  diag(k)
)
theta <- 0.001 * diag(k)
theta_l <- list(
  t(chol(theta))
)
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
)
gamma <- list(
  diag(x = 0.10, nrow = p, ncol = j)
)
kappa <- list(
  diag(x = 0.10, nrow = k, ncol = j)
)

# Type 0
ssm <- SimSSMIVary(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)

plot(ssm)

# Type 1
```

```
ssm <- SimSSMIVary(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
)

plot(ssm)

# Type 2
ssm <- SimSSMIVary(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)

plot(ssm)
```

SimSSMLinGrowth *Simulate Data from the Linear Growth Curve Model*

Description

This function simulates data from the linear growth curve model.

Usage

```
SimSSMLinGrowth(
  n,
```

```
  time,
  mu0,
  sigma0_l,
  theta_l,
  type = 0,
  x = NULL,
  gamma = NULL,
  kappa = NULL
)
```

Arguments

`n`	Positive integer. Number of individuals.
`time`	Positive integer. Number of time points.
`mu0`	Numeric vector. A vector of length two. The first element is the mean of the intercept, and the second element is the mean of the slope.
`sigma0_l`	Numeric matrix. Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of the intercept and the slope.
`theta_l`	Numeric. Square root of the common measurement error variance.
`type`	Integer. State space model type. See Details for more information.
`x`	List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.
`gamma`	Numeric matrix. Matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
`kappa`	Numeric matrix. Matrix linking the covariates to the observed variables at current time point ($\boldsymbol{\kappa}$).

Details

Type 0:

The measurement model is given by

$$Y_{i,t} = \left(\begin{array}{cc} 1 & 0 \end{array} \right) \left(\begin{array}{c} \eta_{0_{i,t}} \\ \eta_{1_{i,t}} \end{array} \right) + \boldsymbol{\varepsilon}_{i,t}, \quad \text{with} \quad \boldsymbol{\varepsilon}_{i,t} \sim \mathcal{N}\left(0, \theta\right)$$

where $Y_{i,t}$, $\eta_{0_{i,t}}$, $\eta_{1_{i,t}}$, and $\boldsymbol{\varepsilon}_{i,t}$ are random variables and θ is a model parameter. $Y_{i,t}$ is the observed random variable at time t and individual i, $\eta_{0_{i,t}}$ (intercept) and $\eta_{1_{i,t}}$ (slope) form a vector of latent random variables at time t and individual i, and $\boldsymbol{\varepsilon}_{i,t}$ a vector of random measurement errors at time t and individual i. θ is the variance of $\boldsymbol{\varepsilon}$.

The dynamic structure is given by

$$\left(\begin{array}{c} \eta_{0_{i,t}} \\ \eta_{1_{i,t}} \end{array} \right) = \left(\begin{array}{cc} 1 & 1 \\ 0 & 1 \end{array} \right) \left(\begin{array}{c} \eta_{0_{i,t-1}} \\ \eta_{1_{i,t-1}} \end{array} \right).$$

The mean vector and covariance matrix of the intercept and slope are captured in the mean vector and covariance matrix of the initial condition given by

$$\boldsymbol{\mu}_{\boldsymbol{\eta}|0} = \left(\begin{array}{c} \mu_{\eta_0} \\ \mu_{\eta_1} \end{array} \right) \quad \text{and,}$$

$$\boldsymbol{\Sigma}_{\boldsymbol{\eta}|0} = \begin{pmatrix} \sigma^2_{\eta_0} & \sigma_{\eta_0, \eta_1} \\ \sigma_{\eta_1, \eta_0} & \sigma^2_{\eta_1} \end{pmatrix}.$$

Type 1:
The measurement model is given by

$$Y_{i,t} = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} \eta_{0_{i,t}} \\ \eta_{1_{i,t}} \end{pmatrix} + \boldsymbol{\varepsilon}_{i,t}, \quad \text{with} \quad \boldsymbol{\varepsilon}_{i,t} \sim \mathcal{N}(0, \theta).$$

The dynamic structure is given by

$$\begin{pmatrix} \eta_{0_{i,t}} \\ \eta_{1_{i,t}} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \eta_{0_{i,t-1}} \\ \eta_{1_{i,t-1}} \end{pmatrix} + \boldsymbol{\Gamma} \mathbf{x}_{i,t}$$

where $\mathbf{x}_{i,t}$ represents a vector of covariates at time t and individual i, and $\boldsymbol{\Gamma}$ the coefficient matrix linking the covariates to the latent variables.

Type 2:
The measurement model is given by

$$Y_{i,t} = \begin{pmatrix} 1 & 0 \end{pmatrix} \begin{pmatrix} \eta_{0_{i,t}} \\ \eta_{1_{i,t}} \end{pmatrix} + \boldsymbol{\kappa} \mathbf{x}_{i,t} + \boldsymbol{\varepsilon}_{i,t}, \quad \text{with} \quad \boldsymbol{\varepsilon}_{i,t} \sim \mathcal{N}(0, \theta)$$

where $\boldsymbol{\kappa}$ represents the coefficient matrix linking the covariates to the observed variables.
The dynamic structure is given by

$$\begin{pmatrix} \eta_{0_{i,t}} \\ \eta_{1_{i,t}} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \eta_{0_{i,t-1}} \\ \eta_{1_{i,t-1}} \end{pmatrix} + \boldsymbol{\Gamma} \mathbf{x}_{i,t}.$$

Value

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

- `call`: Function call.
- `args`: Function arguments.
- `data`: Generated data which is a list of length `n`. Each element of `data` is a list with the following elements:
  - `id`: A vector of ID numbers with length `l`, where `l` is the value of the function argument `time`.
  - `time`: A vector time points of length `l`.
  - `y`: A `l` by `k` matrix of values for the manifest variables.
  - `eta`: A `l` by `p` matrix of values for the latent variables.
  - `x`: A `l` by `j` matrix of values for the covariates (when covariates are included).
- `fun`: Function used.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), PBSSMFixed(), PBSSMLinSDEFixed(), PBSSMOUFixed(), PBSSMVARFixed(), SimBetaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), TestPhi(), TestStability(), TestStationarity()

Examples

```
# prepare parameters
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 5
## dynamic structure
p <- 2
mu0 <- c(0.615, 1.006)
sigma0 <- matrix(
  data = c(
    1.932,
    0.618,
    0.618,
    0.587
  ),
  nrow = p
)
sigma0_l <- t(chol(sigma0))
## measurement model
k <- 1
theta <- 0.50
theta_l <- sqrt(theta)
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    return(
      matrix(
        data = rnorm(n = j * time),
        nrow = j
      )
    )
  }
)
gamma <- diag(x = 0.10, nrow = p, ncol = j)
```

```
kappa <- diag(x = 0.10, nrow = k, ncol = j)

# Type 0
ssm <- SimSSMLinGrowth(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  theta_l = theta_l,
  type = 0
)

plot(ssm)

# Type 1
ssm <- SimSSMLinGrowth(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
)

plot(ssm)

# Type 2
ssm <- SimSSMLinGrowth(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  theta_l = theta_l,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)

plot(ssm)
```

SimSSMLinGrowthIVary *Simulate Data from the Linear Growth Curve Model (Individual-Varying Parameters)*

Description

This function simulates data from the linear growth curve model. It assumes that the parameters can vary across individuals.

Usage

```
SimSSMLinGrowthIVary(
  n,
  time,
  mu0,
  sigma0_l,
  theta_l,
  type = 0,
  x = NULL,
  gamma = NULL,
  kappa = NULL
)
```

Arguments

`n`	Positive integer. Number of individuals.
`time`	Positive integer. Number of time points.
`mu0`	A list of numeric vectors. Each element of the list is a vector of length two. The first element is the mean of the intercept, and the second element is the mean of the slope.
`sigma0_l`	A list of numeric matrices. Each element of the list is the Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of the intercept and the slope.
`theta_l`	A list numeric values. Each element of the list is the square root of the common measurement error variance.
`type`	Integer. State space model type. See Details in `SimSSMLinGrowth()` for more information.
`x`	List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.
`gamma`	List of numeric matrices. Each element of the list is the matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
`kappa`	List of numeric matrices. Each element of the list is the matrix linking the covariates to the observed variables at current time point ($\boldsymbol{\kappa}$).

Details

Parameters can vary across individuals by providing a list of parameter values. If the length of any of the parameters (`mu0`, `sigma0`, `mu`, `theta_l`, `gamma`, or `kappa`) is less the `n`, the function will cycle through the available values.

Value

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

- `call`: Function call.
- `args`: Function arguments.
- `data`: Generated data which is a list of length `n`. Each element of `data` is a list with the following elements:
  - `id`: A vector of ID numbers with length `l`, where `l` is the value of the function argument `time`.
  - `time`: A vector time points of length `l`.
  - `y`: A `l` by `k` matrix of values for the manifest variables.
  - `eta`: A `l` by `p` matrix of values for the latent variables.
  - `x`: A `l` by `j` matrix of values for the covariates (when covariates are included).
- `fun`: Function used.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
# prepare parameters
# In this example, the mean vector of the intercept and slope vary.
# Specifically,
# there are two sets of values representing two latent classes.
set.seed(42)
## number of individuals
n <- 10
## time points
time <- 5
```

```
## dynamic structure
p <- 2
mu0_1 <- c(0.615, 1.006) # lower starting point, higher growth
mu0_2 <- c(1.000, 0.500) # higher starting point, lower growth
mu0 <- list(mu0_1, mu0_2)
sigma0 <- matrix(
  data = c(
    1.932,
    0.618,
    0.618,
    0.587
  ),
  nrow = p
)
sigma0_l <- list(t(chol(sigma0)))
## measurement model
k <- 1
theta <- 0.50
theta_l <- list(sqrt(theta))
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
)
gamma <- list(
  diag(x = 0.10, nrow = p, ncol = j)
)
kappa <- list(
  diag(x = 0.10, nrow = k, ncol = j)
)

# Type 0
ssm <- SimSSMLinGrowthIVary(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  theta_l = theta_l,
  type = 0
)

plot(ssm)

# Type 1
ssm <- SimSSMLinGrowthIVary(
  n = n,
```

```
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
)

plot(ssm)

# Type 2
ssm <- SimSSMLinGrowthIVary(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  theta_l = theta_l,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)

plot(ssm)
```

`SimSSMLinSDEFixed` *Simulate Data from the Linear Stochastic Differential Equation Model using a State Space Model Parameterization (Fixed Parameters)*

Description

This function simulates data from the linear stochastic differential equation model using a state space model parameterization. It assumes that the parameters remain constant across individuals and over time.

Usage

```
SimSSMLinSDEFixed(
  n,
  time,
  delta_t = 1,
  mu0,
  sigma0_l,
  iota,
  phi,
  sigma_l,
```

```
    nu,
    lambda,
    theta_l,
    type = 0,
    x = NULL,
    gamma = NULL,
    kappa = NULL
)
```

Arguments

n	Positive integer. Number of individuals.
time	Positive integer. Number of time points.
delta_t	Numeric. Time interval (Δ_t).
mu0	Numeric vector. Mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta} \mid 0}$).
sigma0_l	Numeric matrix. Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta} \mid 0}$).
iota	Numeric vector. An unobserved term that is constant over time ($\boldsymbol{\iota}$).
phi	Numeric matrix. The drift matrix which represents the rate of change of the solution in the absence of any random fluctuations ($\boldsymbol{\Phi}$).
sigma_l	Numeric matrix. Cholesky factorization (`t(chol(sigma))`) of the covariance matrix of volatility or randomness in the process ($\boldsymbol{\Sigma}$).
nu	Numeric vector. Vector of intercept values for the measurement model ($\boldsymbol{\nu}$).
lambda	Numeric matrix. Factor loading matrix linking the latent variables to the observed variables ($\boldsymbol{\Lambda}$).
theta_l	Numeric matrix. Cholesky factorization (`t(chol(theta))`) of the covariance matrix of the measurement error ($\boldsymbol{\Theta}$).
type	Integer. State space model type. See Details for more information.
x	List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.
gamma	Numeric matrix. Matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
kappa	Numeric matrix. Matrix linking the covariates to the observed variables at current time point ($\boldsymbol{\kappa}$).

Details

Type 0:

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. $\boldsymbol{\nu}$ denotes a vector of intercepts, $\boldsymbol{\Lambda}$ a matrix of factor loadings, and $\boldsymbol{\Theta}$ the covariance matrix of $\boldsymbol{\varepsilon}$.
An alternative representation of the measurement error is given by

$$\boldsymbol{\varepsilon}_{i,t} = \boldsymbol{\Theta}^{\frac{1}{2}} \mathbf{z}_{i,t}, \quad \text{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

where $\mathbf{z}_{i,t}$ is a vector of independent standard normal random variables and $\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)' = \boldsymbol{\Theta}$.
The dynamic structure is given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \left(\boldsymbol{\iota} + \boldsymbol{\Phi}\boldsymbol{\eta}_{i,t}\right)\mathrm{d}t + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}$$

where $\boldsymbol{\iota}$ is a term which is unobserved and constant over time, $\boldsymbol{\Phi}$ is the drift matrix which represents the rate of change of the solution in the absence of any random fluctuations, $\boldsymbol{\Sigma}$ is the matrix of volatility or randomness in the process, and $\mathrm{d}\boldsymbol{W}$ is a Wiener process or Brownian motion, which represents random fluctuations.

Type 1:
The measurement model is given by

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

The dynamic structure is given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \left(\boldsymbol{\iota} + \boldsymbol{\Phi}\boldsymbol{\eta}_{i,t}\right)\mathrm{d}t + \boldsymbol{\Gamma}\mathbf{x}_{i,t} + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}$$

where $\mathbf{x}_{i,t}$ represents a vector of covariates at time t and individual i, and $\boldsymbol{\Gamma}$ the coefficient matrix linking the covariates to the latent variables.

Type 2:
The measurement model is given by

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

where $\boldsymbol{\kappa}$ represents the coefficient matrix linking the covariates to the observed variables.
The dynamic structure is given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \left(\boldsymbol{\iota} + \boldsymbol{\Phi}\boldsymbol{\eta}_{i,t}\right)\mathrm{d}t + \boldsymbol{\Gamma}\mathbf{x}_{i,t} + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}.$$

State Space Parameterization:
The state space parameters as a function of the linear stochastic differential equation model parameters are given by

$$\boldsymbol{\beta}_{\Delta t_{l_i}} = \exp\left(\Delta t \boldsymbol{\Phi}\right)$$

$$\boldsymbol{\alpha}_{\Delta t_{l_i}} = \boldsymbol{\Phi}^{-1}\left(\boldsymbol{\beta} - \mathbf{I}_p\right)\boldsymbol{\iota}$$

$$\mathrm{vec}\left(\boldsymbol{\Psi}_{\Delta t_{l_i}}\right) = \left[\left(\boldsymbol{\Phi} \otimes \mathbf{I}_p\right) + \left(\mathbf{I}_p \otimes \boldsymbol{\Phi}\right)\right]\left[\exp\left(\left[\left(\boldsymbol{\Phi} \otimes \mathbf{I}_p\right) + \left(\mathbf{I}_p \otimes \boldsymbol{\Phi}\right)\right]\Delta t\right) - \mathbf{I}_{p \times p}\right]\mathrm{vec}\left(\boldsymbol{\Sigma}\right)$$

where p is the number of latent variables and Δt is the time interval.

Value

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

- `call`: Function call.
- `args`: Function arguments.
- `data`: Generated data which is a list of length `n`. Each element of `data` is a list with the following elements:
  - `id`: A vector of ID numbers with length `l`, where `l` is the value of the function argument `time`.
  - `time`: A vector time points of length `l`.
  - `y`: A `l` by `k` matrix of values for the manifest variables.
  - `eta`: A `l` by `p` matrix of values for the latent variables.
  - `x`: A `l` by `j` matrix of values for the covariates (when covariates are included).
- `fun`: Function used.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

Chow, S.-M., Losardo, D., Park, J., & Molenaar, P. C. M. (2023). Continuous-time dynamic models: Connections to structural equation models and other discrete-time models. In R. H. Hoyle (Ed.), Handbook of structural equation modeling (2nd ed.). The Guilford Press.

Harvey, A. C. (1990). Forecasting, structural time series models and the Kalman filter. Cambridge University Press. doi:10.1017/cbo9781107049994

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
# prepare parameters
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
delta_t <- 0.10
## dynamic structure
```

```
p <- 2
mu0 <- c(-3.0, 1.5)
sigma0 <- 0.001 * diag(p)
sigma0_l <- t(chol(sigma0))
iota <- c(0.317, 0.230)
phi <- matrix(
  data = c(
    -0.10,
    0.05,
    0.05,
    -0.10
  ),
  nrow = p
)
sigma <- matrix(
  data = c(
    2.79,
    0.06,
    0.06,
    3.27
  ),
  nrow = p
)
sigma_l <- t(chol(sigma))
## measurement model
k <- 2
nu <- rep(x = 0, times = k)
lambda <- diag(k)
theta <- 0.001 * diag(k)
theta_l <- t(chol(theta))
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
)
gamma <- diag(x = 0.10, nrow = p, ncol = j)
kappa <- diag(x = 0.10, nrow = k, ncol = j)

# Type 0
ssm <- SimSSMLinSDEFixed(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  iota = iota,
```

```
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)

plot(ssm)

# Type 1
ssm <- SimSSMLinSDEFixed(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  iota = iota,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
)

plot(ssm)

# Type 2
ssm <- SimSSMLinSDEFixed(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  iota = iota,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)

plot(ssm)
```

SimSSMLinSDEIVary — *Simulate Data from the Linear Stochastic Differential Equation Model using a State Space Model Parameterization (Individual-Varying Parameters)*

Description

This function simulates data from the linear stochastic differential equation model using a state space model parameterization. It assumes that the parameters can vary across individuals.

Usage

```
SimSSMLinSDEIVary(
  n,
  time,
  delta_t = 1,
  mu0,
  sigma0_l,
  iota,
  phi,
  sigma_l,
  nu,
  lambda,
  theta_l,
  type = 0,
  x = NULL,
  gamma = NULL,
  kappa = NULL
)
```

Arguments

`n`	Positive integer. Number of individuals.
`time`	Positive integer. Number of time points.
`delta_t`	Numeric. Time interval. The default value is `1.0` with an option to use a numeric value for the discretized state space model parameterization of the linear stochastic differential equation model.
`mu0`	List of numeric vectors. Each element of the list is the mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta}\mid 0}$).
`sigma0_l`	List of numeric matrices. Each element of the list is the Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta}\mid 0}$).
`iota`	List of numeric vectors. Each element of the list is an unobserved term that is constant over time ($\boldsymbol{\iota}$).

phi	List of numeric matrix. Each element of the list is the drift matrix which represents the rate of change of the solution in the absence of any random fluctuations ($\boldsymbol{\Phi}$).
sigma_l	List of numeric matrix. Each element of the list is the Cholesky factorization (`t(chol(sigma))`) of the covariance matrix of volatility or randomness in the process $\boldsymbol{\Sigma}$.
nu	List of numeric vectors. Each element of the list is the vector of intercept values for the measurement model ($\boldsymbol{\nu}$).
lambda	List of numeric matrices. Each element of the list is the factor loading matrix linking the latent variables to the observed variables ($\boldsymbol{\Lambda}$).
theta_l	List of numeric matrices. Each element of the list is the Cholesky factorization (`t(chol(theta))`) of the covariance matrix of the measurement error ($\boldsymbol{\Theta}$).
type	Integer. State space model type. See Details in `SimSSMLinSDEFixed()` for more information.
x	List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.
gamma	List of numeric matrices. Each element of the list is the matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
kappa	List of numeric matrices. Each element of the list is the matrix linking the covariates to the observed variables at current time point ($\boldsymbol{\kappa}$).

Details

Parameters can vary across individuals by providing a list of parameter values. If the length of any of the parameters (`mu0`, `sigma0_l`, `iota`, `phi`, `sigma_l`, `nu`, `lambda`, `theta_l`, `gamma`, or `kappa`) is less the `n`, the function will cycle through the available values.

Value

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

- `call`: Function call.
- `args`: Function arguments.
- `data`: Generated data which is a list of length `n`. Each element of `data` is a list with the following elements:
  - `id`: A vector of ID numbers with length `l`, where `l` is the value of the function argument `time`.
  - `time`: A vector time points of length `l`.
  - `y`: A `l` by `k` matrix of values for the manifest variables.
  - `eta`: A `l` by `p` matrix of values for the latent variables.
  - `x`: A `l` by `j` matrix of values for the covariates (when covariates are included).
- `fun`: Function used.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

Chow, S.-M., Losardo, D., Park, J., & Molenaar, P. C. M. (2023). Continuous-time dynamic models: Connections to structural equation models and other discrete-time models. In R. H. Hoyle (Ed.), Handbook of structural equation modeling (2nd ed.). The Guilford Press.

Harvey, A. C. (1990). Forecasting, structural time series models and the Kalman filter. Cambridge University Press. doi:10.1017/cbo9781107049994

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
# prepare parameters
# In this example, phi varies across individuals.
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
delta_t <- 0.10
## dynamic structure
p <- 2
mu0 <- list(
  c(-3.0, 1.5)
)
sigma0 <- 0.001 * diag(p)
sigma0_l <- list(
  t(chol(sigma0))
)
iota <- list(
  c(0.317, 0.230)
)
phi <- list(
  -0.1 * diag(p),
  -0.2 * diag(p),
  -0.3 * diag(p),
  -0.4 * diag(p),
  -0.5 * diag(p)
```

```
)
sigma <- matrix(
  data = c(
    2.79,
    0.06,
    0.06,
    3.27
  ),
  nrow = p
)
sigma_l <- list(
  t(chol(sigma))
)
## measurement model
k <- 2
nu <- list(
  rep(x = 0, times = k)
)
lambda <- list(
  diag(k)
)
theta <- 0.001 * diag(k)
theta_l <- list(
  t(chol(theta))
)
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
)
gamma <- list(
  diag(x = 0.10, nrow = p, ncol = j)
)
kappa <- list(
  diag(x = 0.10, nrow = k, ncol = j)
)

# Type 0
ssm <- SimSSMLinSDEIVary(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  iota = iota,
  phi = phi,
```

```
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)

plot(ssm)

# Type 1
ssm <- SimSSMLinSDEIVary(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  iota = iota,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
)

plot(ssm)

# Type 2
ssm <- SimSSMLinSDEIVary(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  iota = iota,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)

plot(ssm)
```

`SimSSMOUFixed` *Simulate Data from the Ornstein–Uhlenbeck Model using a State Space Model Parameterization (Fixed Parameters)*

Description

This function simulates data from the Ornstein–Uhlenbeck (OU) model using a state space model parameterization. It assumes that the parameters remain constant across individuals and over time.

Usage

```
SimSSMOUFixed(
  n,
  time,
  delta_t = 1,
  mu0,
  sigma0_l,
  mu,
  phi,
  sigma_l,
  nu,
  lambda,
  theta_l,
  type = 0,
  x = NULL,
  gamma = NULL,
  kappa = NULL
)
```

Arguments

`n`	Positive integer. Number of individuals.
`time`	Positive integer. Number of time points.
`delta_t`	Numeric. Time interval (Δ_t).
`mu0`	Numeric vector. Mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta}\mid 0}$).
`sigma0_l`	Numeric matrix. Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta}\mid 0}$).
`mu`	Numeric vector. The long-term mean or equilibrium level ($\boldsymbol{\mu}$).
`phi`	Numeric matrix. The drift matrix which represents the rate of change of the solution in the absence of any random fluctuations ($\boldsymbol{\Phi}$). It also represents the rate of mean reversion, determining how quickly the variable returns to its mean.
`sigma_l`	Numeric matrix. Cholesky factorization (`t(chol(sigma))`) of the covariance matrix of volatility or randomness in the process ($\boldsymbol{\Sigma}$).

nu	Numeric vector. Vector of intercept values for the measurement model ($\boldsymbol{\nu}$).
lambda	Numeric matrix. Factor loading matrix linking the latent variables to the observed variables ($\boldsymbol{\Lambda}$).
theta_l	Numeric matrix. Cholesky factorization (`t(chol(theta))`) of the covariance matrix of the measurement error ($\boldsymbol{\Theta}$).
type	Integer. State space model type. See Details for more information.
x	List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.
gamma	Numeric matrix. Matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
kappa	Numeric matrix. Matrix linking the covariates to the observed variables at current time point ($\boldsymbol{\kappa}$).

Details

Type 0:

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. $\boldsymbol{\nu}$ denotes a vector of intercepts, $\boldsymbol{\Lambda}$ a matrix of factor loadings, and $\boldsymbol{\Theta}$ the covariance matrix of $\boldsymbol{\varepsilon}$.

An alternative representation of the measurement error is given by

$$\boldsymbol{\varepsilon}_{i,t} = \boldsymbol{\Theta}^{\frac{1}{2}}\mathbf{z}_{i,t}, \quad \text{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

where $\mathbf{z}_{i,t}$ is a vector of independent standard normal random variables and $\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)\left(\boldsymbol{\Theta}^{\frac{1}{2}}\right)' = \boldsymbol{\Theta}$.

The dynamic structure is given by

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

where $\boldsymbol{\mu}$ is the long-term mean or equilibrium level, $\boldsymbol{\Phi}$ is the rate of mean reversion, determining how quickly the variable returns to its mean, $\boldsymbol{\Sigma}$ is the matrix of volatility or randomness in the process, and $\mathrm{d}\boldsymbol{W}$ is a Wiener process or Brownian motion, which represents random fluctuations.

Type 1:

The measurement model is given by

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

The dynamic structure is given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \boldsymbol{\Phi}\left(\boldsymbol{\eta}_{i,t} - \boldsymbol{\mu}\right)\mathrm{d}t + \boldsymbol{\Gamma}\mathbf{x}_{i,t} + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}$$

where $\mathbf{x}_{i,t}$ represents a vector of covariates at time t and individual i, and $\boldsymbol{\Gamma}$ the coefficient matrix linking the covariates to the latent variables.

Type 2:
The measurement model is given by

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

where $\boldsymbol{\kappa}$ represents the coefficient matrix linking the covariates to the observed variables. The dynamic structure is given by

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \boldsymbol{\Phi}\left(\boldsymbol{\eta}_{i,t} - \boldsymbol{\mu}\right)\mathrm{d}t + \boldsymbol{\Gamma}\mathbf{x}_{i,t} + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}.$$

The OU model as a linear stochastic differential equation model:
The OU model is a first-order linear stochastic differential equation model in the form of

$$\mathrm{d}\boldsymbol{\eta}_{i,t} = \left(\boldsymbol{\iota} + \boldsymbol{\Phi}\boldsymbol{\eta}_{i,t}\right)\mathrm{d}t + \boldsymbol{\Sigma}^{\frac{1}{2}}\mathrm{d}\mathbf{W}_{i,t}$$

where $\boldsymbol{\mu} = -\boldsymbol{\Phi}^{-1}\boldsymbol{\iota}$ and, equivalently $\boldsymbol{\iota} = -\boldsymbol{\Phi}\boldsymbol{\mu}$.

Value

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

- `call`: Function call.
- `args`: Function arguments.
- `data`: Generated data which is a list of length `n`. Each element of `data` is a list with the following elements:
  - `id`: A vector of ID numbers with length `l`, where `l` is the value of the function argument `time`.
  - `time`: A vector time points of length `l`.
  - `y`: A `l` by `k` matrix of values for the manifest variables.
  - `eta`: A `l` by `p` matrix of values for the latent variables.
  - `x`: A `l` by `j` matrix of values for the covariates (when covariates are included).
- `fun`: Function used.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

Chow, S.-M., Losardo, D., Park, J., & Molenaar, P. C. M. (2023). Continuous-time dynamic models: Connections to structural equation models and other discrete-time models. In R. H. Hoyle (Ed.), Handbook of structural equation modeling (2nd ed.). The Guilford Press.

Harvey, A. C. (1990). Forecasting, structural time series models and the Kalman filter. Cambridge University Press. doi:10.1017/cbo9781107049994

Oravecz, Z., Tuerlinckx, F., & Vandekerckhove, J. (2011). A hierarchical latent stochastic differential equation model for affective dynamics. Psychological Methods, 16 (4), 468–490. doi:10.1037/a0024375

Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the theory of the brownian motion. Physical Review, 36 (5), 823–841. doi:10.1103/physrev.36.823

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
# prepare parameters
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
delta_t <- 0.10
## dynamic structure
p <- 2
mu0 <- c(-3.0, 1.5)
sigma0 <- 0.001 * diag(p)
sigma0_l <- t(chol(sigma0))
mu <- c(5.76, 5.18)
phi <- matrix(
  data = c(
    -0.10,
    0.05,
    0.05,
    -0.10
  ),
  nrow = p
)
sigma <- matrix(
  data = c(
    2.79,
    0.06,
    0.06,
    3.27
  ),
  nrow = p
)
sigma_l <- t(chol(sigma))
## measurement model
k <- 2
nu <- rep(x = 0, times = k)
lambda <- diag(k)
theta <- 0.001 * diag(k)
```

```
theta_l <- t(chol(theta))
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
)
gamma <- diag(x = 0.10, nrow = p, ncol = j)
kappa <- diag(x = 0.10, nrow = k, ncol = j)

# Type 0
ssm <- SimSSMOUFixed(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  mu = mu,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)

plot(ssm)

# Type 1
ssm <- SimSSMOUFixed(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  mu = mu,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
)

plot(ssm)
```

```
# Type 2
ssm <- SimSSMOUFixed(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  mu = mu,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)

plot(ssm)
```

`SimSSMOUIVary` *Simulate Data from the Ornstein–Uhlenbeck Model using a State Space Model Parameterization (Individual-Varying Parameters)*

Description

This function simulates data from the Ornstein–Uhlenbeck model using a state space model parameterization. It assumes that the parameters can vary across individuals.

Usage

```
SimSSMOUIVary(
  n,
  time,
  delta_t = 1,
  mu0,
  sigma0_l,
  mu,
  phi,
  sigma_l,
  nu,
  lambda,
  theta_l,
  type = 0,
  x = NULL,
  gamma = NULL,
```

```
  kappa = NULL
)
```

Arguments

- `n` — Positive integer. Number of individuals.
- `time` — Positive integer. Number of time points.
- `delta_t` — Numeric. Time interval. The default value is `1.0` with an option to use a numeric value for the discretized state space model parameterization of the linear stochastic differential equation model.
- `mu0` — List of numeric vectors. Each element of the list is the mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta}|0}$).
- `sigma0_l` — List of numeric matrices. Each element of the list is the Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta}|0}$).
- `mu` — List of numeric vectors. Each element of the list is the long-term mean or equilibrium level ($\boldsymbol{\mu}$).
- `phi` — List of numeric matrix. Each element of the list is the drift matrix which represents the rate of change of the solution in the absence of any random fluctuations ($\boldsymbol{\Phi}$). It also represents the rate of mean reversion, determining how quickly the variable returns to its mean.
- `sigma_l` — List of numeric matrix. Each element of the list is the Cholesky factorization (`t(chol(sigma))`) of the covariance matrix of volatility or randomness in the process $\boldsymbol{\Sigma}$.
- `nu` — List of numeric vectors. Each element of the list is the vector of intercept values for the measurement model ($\boldsymbol{\nu}$).
- `lambda` — List of numeric matrices. Each element of the list is the factor loading matrix linking the latent variables to the observed variables ($\boldsymbol{\Lambda}$).
- `theta_l` — List of numeric matrices. Each element of the list is the Cholesky factorization (`t(chol(theta))`) of the covariance matrix of the measurement error ($\boldsymbol{\Theta}$).
- `type` — Integer. State space model type. See Details in `SimSSMOUFixed()` for more information.
- `x` — List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.
- `gamma` — List of numeric matrices. Each element of the list is the matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).
- `kappa` — List of numeric matrices. Each element of the list is the matrix linking the covariates to the observed variables at current time point ($\boldsymbol{\kappa}$).

Details

Parameters can vary across individuals by providing a list of parameter values. If the length of any of the parameters (`mu0`, `sigma0_l`, `mu`, `phi`, `sigma_l`, `nu`, `lambda`, `theta_l`, `gamma`, or `kappa`) is less the `n`, the function will cycle through the available values.

Value

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

- `call`: Function call.
- `args`: Function arguments.
- `data`: Generated data which is a list of length `n`. Each element of `data` is a list with the following elements:
  - `id`: A vector of ID numbers with length `l`, where `l` is the value of the function argument `time`.
  - `time`: A vector time points of length `l`.
  - `y`: A `l` by `k` matrix of values for the manifest variables.
  - `eta`: A `l` by `p` matrix of values for the latent variables.
  - `x`: A `l` by `j` matrix of values for the covariates (when covariates are included).
- `fun`: Function used.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

Chow, S.-M., Losardo, D., Park, J., & Molenaar, P. C. M. (2023). Continuous-time dynamic models: Connections to structural equation models and other discrete-time models. In R. H. Hoyle (Ed.), Handbook of structural equation modeling (2nd ed.). The Guilford Press.

Harvey, A. C. (1990). Forecasting, structural time series models and the Kalman filter. Cambridge University Press. doi:10.1017/cbo9781107049994

Oravecz, Z., Tuerlinckx, F., & Vandekerckhove, J. (2011). A hierarchical latent stochastic differential equation model for affective dynamics. Psychological Methods, 16 (4), 468–490. doi:10.1037/a0024375

Uhlenbeck, G. E., & Ornstein, L. S. (1930). On the theory of the brownian motion. Physical Review, 36 (5), 823–841. doi:10.1103/physrev.36.823

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
# prepare parameters
# In this example, phi varies across individuals.
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
delta_t <- 0.10
## dynamic structure
p <- 2
mu0 <- list(
  c(-3.0, 1.5)
)
sigma0 <- 0.001 * diag(p)
sigma0_l <- list(
  t(chol(sigma0))
)
mu <- list(
  c(5.76, 5.18)
)
phi <- list(
  -0.1 * diag(p),
  -0.2 * diag(p),
  -0.3 * diag(p),
  -0.4 * diag(p),
  -0.5 * diag(p)
)
sigma <- matrix(
  data = c(
    2.79,
    0.06,
    0.06,
    3.27
  ),
  nrow = p
)
sigma_l <- list(
  t(chol(sigma))
)
## measurement model
k <- 2
nu <- list(
  rep(x = 0, times = k)
)
lambda <- list(
  diag(k)
)
theta <- 0.001 * diag(k)
theta_l <- list(
  t(chol(theta))
)
```

```
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
)
gamma <- list(
  diag(x = 0.10, nrow = p, ncol = j)
)
kappa <- list(
  diag(x = 0.10, nrow = k, ncol = j)
)

# Type 0
ssm <- SimSSMOUIVary(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  mu = mu,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)

plot(ssm)

# Type 1
ssm <- SimSSMOUIVary(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  mu = mu,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
```

```
)

plot(ssm)

# Type 2
ssm <- SimSSMOUIVary(
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  mu = mu,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)

plot(ssm)
```

SimSSMVARFixed *Simulate Data from the Vector Autoregressive Model (Fixed Parameters)*

Description

This function simulates data from the vector autoregressive model using a state space model parameterization. It assumes that the parameters remain constant across individuals and over time.

Usage

```
SimSSMVARFixed(
  n,
  time,
  mu0,
  sigma0_l,
  alpha,
  beta,
  psi_l,
  type = 0,
  x = NULL,
  gamma = NULL
)
```

Arguments

`n`	Positive integer. Number of individuals.
`time`	Positive integer. Number of time points.
`mu0`	Numeric vector. Mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta}\|0}$).
`sigma0_l`	Numeric matrix. Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta}\|0}$).
`alpha`	Numeric vector. Vector of constant values for the dynamic model ($\boldsymbol{\alpha}$).
`beta`	Numeric matrix. Transition matrix relating the values of the latent variables at the previous to the current time point ($\boldsymbol{\beta}$).
`psi_l`	Numeric matrix. Cholesky factorization (`t(chol(psi))`) of the covariance matrix of the process noise ($\boldsymbol{\Psi}$).
`type`	Integer. State space model type. See Details for more information.
`x`	List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.
`gamma`	Numeric matrix. Matrix linking the covariates to the latent variables at current time point ($\boldsymbol{\Gamma}$).

Details

Type 0:

The measurement model is given by

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

where $\mathbf{y}_{i,t}$ represents a vector of observed variables and $\boldsymbol{\eta}_{i,t}$ a vector of latent variables for individual i and time t. Since the observed and latent variables are equal, we only generate data from the dynamic structure.

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 $\boldsymbol{\zeta}_{i,t}$.

An alternative representation of the dynamic noise is given by

$$\boldsymbol{\zeta}_{i,t} = \boldsymbol{\Psi}^{\frac{1}{2}}\mathbf{z}_{i,t}, \quad \text{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

where $\left(\boldsymbol{\Psi}^{\frac{1}{2}}\right)\left(\boldsymbol{\Psi}^{\frac{1}{2}}\right)' = \boldsymbol{\Psi}$.

Type 1:

The measurement model is given by

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

The dynamic structure is given by

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

where $\mathbf{x}_{i,t}$ represents a vector of covariates at time t and individual i, and $\boldsymbol{\Gamma}$ the coefficient matrix linking the covariates to the latent variables.

Value

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

- `call`: Function call.
- `args`: Function arguments.
- `data`: Generated data which is a list of length `n`. Each element of `data` is a list with the following elements:
  - `id`: A vector of ID numbers with length `l`, where `l` is the value of the function argument `time`.
  - `time`: A vector time points of length `l`.
  - `y`: A `l` by `k` matrix of values for the manifest variables.
  - `eta`: A `l` by `p` matrix of values for the latent variables.
  - `x`: A `l` by `j` matrix of values for the covariates (when covariates are included).
- `fun`: Function used.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`, `TestStationarity()`

Examples

```
# prepare parameters
set.seed(42)
## number of individuals
n <- 5
## time points
time <- 50
```

```
## dynamic structure
p <- 3
mu0 <- rep(x = 0, times = p)
sigma0 <- 0.001 * diag(p)
sigma0_l <- t(chol(sigma0))
alpha <- rep(x = 0, times = p)
beta <- 0.50 * diag(p)
psi <- 0.001 * diag(p)
psi_l <- t(chol(psi))
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
)
gamma <- diag(x = 0.10, nrow = p, ncol = j)

# Type 0
ssm <- SimSSMVARFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  type = 0
)

plot(ssm)

# Type 1
ssm <- SimSSMVARFixed(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  type = 1,
  x = x,
  gamma = gamma
)

plot(ssm)
```

`SimSSMVARIVary` *Simulate Data from the Vector Autoregressive Model (Individual-Varying Parameters)*

Description

This function simulates data from the vector autoregressive model using a state space model parameterization. It assumes that the parameters can vary across individuals.

Usage

```
SimSSMVARIVary(
  n,
  time,
  mu0,
  sigma0_l,
  alpha,
  beta,
  psi_l,
  type = 0,
  x = NULL,
  gamma = NULL
)
```

Arguments

`n`	Positive integer. Number of individuals.
`time`	Positive integer. Number of time points.
`mu0`	List of numeric vectors. Each element of the list is the mean of initial latent variable values ($\boldsymbol{\mu}_{\boldsymbol{\eta}\mid 0}$).
`sigma0_l`	List of numeric matrices. Each element of the list is the Cholesky factorization (`t(chol(sigma0))`) of the covariance matrix of initial latent variable values ($\boldsymbol{\Sigma}_{\boldsymbol{\eta}\mid 0}$).
`alpha`	List of numeric vectors. Each element of the list is the vector of constant values for the dynamic model ($\boldsymbol{\alpha}$).
`beta`	List of numeric matrices. Each element of the list is the transition matrix relating the values of the latent variables at the previous to the current time point ($\boldsymbol{\beta}$).
`psi_l`	List of numeric matrices. Each element of the list is the Cholesky factorization (`t(chol(psi))`) of the covariance matrix of the process noise ($\boldsymbol{\Psi}$).
`type`	Integer. State space model type. See Details in `SimSSMVARFixed()` for more information.
`x`	List. Each element of the list is a matrix of covariates for each individual `i` in `n`. The number of columns in each matrix should be equal to `time`.

gamma — List of numeric matrices. Each element of the list is the matrix linking the covariates to the latent variables at current time point ($\mathbf{\Gamma}$).

Details

Parameters can vary across individuals by providing a list of parameter values. If the length of any of the parameters (mu0, sigma0_l, alpha, beta, psi_l, gamma, or kappa) is less the n, the function will cycle through the available values.

Value

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

- call: Function call.
- args: Function arguments.
- data: Generated data which is a list of length n. Each element of data is a list with the following elements:
  - id: A vector of ID numbers with length l, where l is the value of the function argument time.
  - time: A vector time points of length l.
  - y: A l by k matrix of values for the manifest variables.
  - eta: A l by p matrix of values for the latent variables.
  - x: A l by j matrix of values for the covariates (when covariates are included).
- fun: Function used.

Author(s)

Ivan Jacob Agaloos Pesigan

References

Chow, S.-M., Ho, M. R., Hamaker, E. L., & Dolan, C. V. (2010). Equivalence and differences between structural equation modeling and state-space modeling techniques. *Structural Equation Modeling: A Multidisciplinary Journal*, 17(2), 303–332. doi:10.1080/10705511003661553

See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), PBSSMFixed(), PBSSMLinSDEFixed(), PBSSMOUFixed(), PBSSMVARFixed(), SimBetaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), TestPhi(), TestStability(), TestStationarity()

Examples

```
# prepare parameters
# In this example, beta varies across individuals.
set.seed(42)
## number of individuals
```

```
n <- 5
## time points
time <- 50
## dynamic structure
p <- 3
mu0 <- list(
  rep(x = 0, times = p)
)
sigma0 <- 0.001 * diag(p)
sigma0_l <- list(
  t(chol(sigma0))
)
alpha <- list(
  rep(x = 0, times = p)
)
beta <- list(
  0.1 * diag(p),
  0.2 * diag(p),
  0.3 * diag(p),
  0.4 * diag(p),
  0.5 * diag(p)
)
psi <- 0.001 * diag(p)
psi_l <- list(
  t(chol(psi))
)
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
)
gamma <- list(
  diag(x = 0.10, nrow = p, ncol = j)
)

# Type 0
ssm <- SimSSMVARIVary(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  type = 0
)
```

```
plot(ssm)

# Type 1
ssm <- SimSSMVARIVary(
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_l = sigma0_l,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  type = 1,
  x = x,
  gamma = gamma
)

plot(ssm)
```

`summary.statespacepb` *Summary Method for an Object of Class* `statespacepb`

Description

Summary Method for an Object of Class `statespacepb`

Usage

```
## S3 method for class 'statespacepb'
summary(object, alpha = NULL, type = "pc", digits = 4, ...)
```

Arguments

`object`	Object of Class `statespacepb`.
`alpha`	Numeric vector. Significance level α. If `alpha = NULL`, use the argument `alpha` used in `object`.
`type`	Charater string. Confidence interval type, that is, `type = "pc"` for percentile; `type = "bc"` for bias corrected.
`digits`	Digits to print.
`...`	additional arguments.

Value

Returns a matrix of estimates, standard errors, number of bootstrap replications, and confidence intervals.

Author(s)

Ivan Jacob Agaloos Pesigan

TestPhi *Test the Drift Matrix*

Description

Both have to be true for the function to return TRUE.

- Test that the real part of all eigenvalues of $\boldsymbol{\Phi}$ are less than zero.
- Test that the diagonal values of $\boldsymbol{\Phi}$ are between 0 to negative inifinity.

Usage

```
TestPhi(phi)
```

Arguments

phi	Numeric matrix. The drift matrix ($\boldsymbol{\Phi}$).

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), PBSSMFixed(), PBSSMLinSDEFixed(), PBSSMOUFixed(), PBSSMVARFixed(), SimBetaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), TestStability(), TestStationarity()

Examples

```
phi <- matrix(
  data = c(
    -0.357, 0.771, -0.450,
    0.0, -0.511, 0.729,
    0, 0, -0.693
  ),
  nrow = 3
)
TestPhi(phi = phi)
```

TestStability — *Test Stability*

Description

The function computes the eigenvalues of the input matrix `x`. It checks if the real part of all eigenvalues is negative. If all eigenvalues have negative real parts, the system is considered stable.

Usage

```
TestStability(x)
```

Arguments

`x`	Numeric matrix.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStationarity()`

Examples

```
x <- matrix(
  data = c(
    -0.357, 0.771, -0.450,
    0.0, -0.511, 0.729,
    0, 0, -0.693
  ),
  nrow = 3
)
TestStability(x)
```

`TestStationarity` *Test Stationarity*

Description

The function computes the eigenvalues of the input matrix `x`. It checks if all eigenvalues have moduli less than 1. If all eigenvalues have moduli less than 1, the system is considered stationary.

Usage

```
TestStationarity(x)
```

Arguments

`x`	Numeric matrix.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

Other Simulation of State Space Models Data Functions: `LinSDE2SSM()`, `PBSSMFixed()`, `PBSSMLinSDEFixed()`, `PBSSMOUFixed()`, `PBSSMVARFixed()`, `SimBetaN()`, `SimPhiN()`, `SimSSMFixed()`, `SimSSMIVary()`, `SimSSMLinGrowth()`, `SimSSMLinGrowthIVary()`, `SimSSMLinSDEFixed()`, `SimSSMLinSDEIVary()`, `SimSSMOUFixed()`, `SimSSMOUIVary()`, `SimSSMVARFixed()`, `SimSSMVARIVary()`, `TestPhi()`, `TestStability()`

Examples

```
x <- matrix(
  data = c(0.5, 0.3, 0.2, 0.4),
  nrow = 2
)
TestStationarity(x)

x <- matrix(
  data = c(0.9, -0.5, 0.8, 0.7),
  nrow = 2
)
TestStationarity(x)
```

`vcov.statespacepb` *Sampling Variance-Covariance Matrix Method for an Object of Class* `statespacepb`

Description

Sampling Variance-Covariance Matrix Method for an Object of Class `statespacepb`

Usage

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

Arguments

`object`	Object of Class `statespacepb`.
`...`	additional arguments.

Value

Returns the variance-covariance matrix of estimates.

Author(s)

Ivan Jacob Agaloos Pesigan

Index