# Package 'simStateSpace'

August 15, 2025
Title Simulate Data from State Space Models
Version 1.2.11
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="">.</doi:10.1080>
<pre>URL https://github.com/jeksterslab/simStateSpace,</pre>
https://jeksterslab.github.io/simStateSpace/
<pre>BugReports https://github.com/jeksterslab/simStateSpace/issues</pre>
License GPL (>= 3)
Encoding UTF-8
<b>Roxygen</b> list(markdown = TRUE)
<b>Depends</b> R (>= $3.5.0$ )
LinkingTo Rcpp, RcppArmadillo
Imports Rcpp, stats
Suggests knitr, rmarkdown, testthat, expm
RoxygenNote 7.3.2
NeedsCompilation yes
Author Ivan Jacob Agaloos Pesigan [aut, cre, cph] (ORCID: <a href="https://orcid.org/0000-0003-4818-8420">https://orcid.org/0000-0003-4818-8420</a> )
Maintainer Ivan Jacob Agaloos Pesigan <r.jeksterslab@gmail.com></r.jeksterslab@gmail.com>
Contents
as.data.frame.simstatespace

LinSDECovY	 11
LinSDEMeanEta	 12
LinSDEMeanY	 13
plot.simstatespace	 14
print.simstatespace	 17
ProjectToStability	 19
SimAlphaN	 21
SimBetaN	 22
SimBetaN2	 23
SimCovDiagN	 24
SimCovN	 25
SimIotaN	 26
SimPhiN	 27
SimSSMFixed	 28
SimSSMIVary	 33
SimSSMLinGrowth	 37
SimSSMLinGrowthIVary	 41
SimSSMLinSDEFixed	 45
SimSSMLinSDEIVary	 51
SimSSMOUFixed	 55
SimSSMOUIVary	 61
SimSSMVARFixed	 66
SimSSMVARIVary	 69
SpectralRadius	 73
SSMCovEta	 74
SSMCovY	 75
SSMMeanEta	 76
SSMMeanY	 77
TestPhi	 79
TestStability	 80
TestStationarity	 81
Index	82

 $\verb"as.data.frame.simstatespace"$ 

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

```
# 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)</pre>
sigma0_l <- t(chol(sigma0))</pre>
alpha < - rep(x = 0, times = p)
beta <- 0.50 * diag(p)
psi <- diag(p)</pre>
psi_l <- t(chol(psi))</pre>
## measurement model
k <- 3
nu \leftarrow rep(x = 0, times = k)
lambda <- diag(k)</pre>
theta <- 0.50 * diag(k)
theta_l <- t(chol(theta))</pre>
## covariates
```

```
j <- 2
x <- lapply(
 X = seq_len(n),
 FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
gamma \leftarrow diag(x = 0.10, nrow = p, ncol = j)
kappa \leftarrow diag(x = 0.10, nrow = k, ncol = j)
# Type 0
ssm <- SimSSMFixed(</pre>
 n = n,
 time = time,
 mu0 = mu0,
  sigma0_1 = sigma0_1,
  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(</pre>
 n = n,
 time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_1 = theta_1,
  type = 1,
  x = x,
  gamma = gamma
)
head(as.data.frame(ssm))
head(as.data.frame(ssm, long = FALSE))
# Type 2
```

as.matrix.simstatespace 5

```
ssm <- SimSSMFixed(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  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

```
# 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)</pre>
sigma0_1 \leftarrow t(chol(sigma0))
alpha <- rep(x = 0, times = p)
beta \leftarrow 0.50 * diag(p)
psi <- diag(p)</pre>
psi_l <- t(chol(psi))</pre>
## measurement model
k <- 3
nu \leftarrow rep(x = 0, times = k)
lambda <- diag(k)</pre>
theta < 0.50 * diag(k)
theta_l <- t(chol(theta))</pre>
## covariates
j <- 2
x <- lapply(
 X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
  }
gamma \leftarrow diag(x = 0.10, nrow = p, ncol = j)
kappa \leftarrow diag(x = 0.10, nrow = k, ncol = j)
# Type 0
ssm <- SimSSMFixed(</pre>
  n = n
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)
```

LinSDE2SSM 7

```
head(as.matrix(ssm))
head(as.matrix(ssm, long = FALSE))
# Type 1
ssm <- SimSSMFixed(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  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(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_1 = theta_1,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)
head(as.matrix(ssm))
head(as.matrix(ssm, long = FALSE))
```

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

8 LinSDE2SSM

## **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 ( $\iota$ ).
phi	Numeric matrix. The drift matrix which represents the rate of change of the solution in the absence of any random fluctuations $(\Phi)$ .
sigma_l	Numeric matrix. Cholesky factorization (t(chol(sigma))) of the covariance matrix of volatility or randomness in the process $(\Sigma)$ .
delta_t	Numeric. Time interval ( $\Delta_t$ ).

#### **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.

$$oldsymbol{\eta}_{i,t_{l_i}} = oldsymbol{lpha}_{\Delta t_{l_i}} + oldsymbol{eta}_{\Delta t_{l_i}} oldsymbol{\eta}_{i,t_{l_i-1}} + oldsymbol{\zeta}_{i,t_{l_i}}, \quad ext{with} \quad oldsymbol{\zeta}_{i,t_{l_i}} \sim \mathcal{N}\left(oldsymbol{0}, oldsymbol{\Psi}_{\Delta t_{l_i}}
ight)$$

with

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

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

$$\operatorname{vec}\left(\mathbf{\Psi}_{\Delta t_{l_{i}}}\right) = \left[\left(\mathbf{\Phi} \otimes \mathbf{I}_{p}\right) + \left(\mathbf{I}_{p} \otimes \mathbf{\Phi}\right)\right] \left[\exp\left(\left[\left(\mathbf{\Phi} \otimes \mathbf{I}_{p}\right) + \left(\mathbf{I}_{p} \otimes \mathbf{\Phi}\right)\right] \Delta t\right) - \mathbf{I}_{p \times p}\right] \operatorname{vec}\left(\mathbf{\Sigma}\right)$$

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

#### Value

Returns a list of state space parameters:

- alpha: Numeric vector. Vector of constant values for the dynamic model  $(\alpha)$ .
- beta: Numeric matrix. Transition matrix relating the values of the latent variables from the previous time point to the current time point.  $(\beta)$ .
- psi\_1: Numeric matrix. Cholesky factorization (t(chol(psi))) of the process noise covariance matrix  $\Psi$ .

LinSDE2SSM 9

## 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: LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

```
p <- 2
iota <- c(0.317, 0.230)
phi <- matrix(</pre>
  data = c(
   -0.10,
   0.05,
   0.05,
   -0.10
 ),
 nrow = p
)
sigma <- matrix(</pre>
  data = c(
    2.79,
    0.06,
    0.06,
    3.27
  ),
  nrow = p
sigma_l <- t(chol(sigma))</pre>
delta_t <- 0.10
LinSDE2SSM(
  iota = iota,
  phi = phi,
  sigma_l = sigma_l,
  delta_t = delta_t
)
```

10 LinSDECovEta

LinSDECovEta	Steady-State Covariance Matrix for the Latent Variables in the Linear
	Stochastic Differential Equation Model

#### **Description**

The steady-state covariance matrix for the latent variables in the linear stochastic differential equation model  $Cov(\eta)$  is the solution to the Sylvester equation, i.e.

$$AX + XB + C = 0,$$

where X is unknown,  $A = \Phi$ ,  $B = \Phi'$ , and  $C = \Sigma$  where  $\Phi$  is the drift matrix and  $\Sigma$  is the covariance matrix of volatility or randomness in the process.

## Usage

```
LinSDECovEta(phi, sigma)
```

## **Arguments**

phi Numeric matrix. The drift matrix which represents the rate of change of the

solution in the absence of any random fluctuations ( $\Phi$ ).

sigma Numeric matrix. The covariance matrix of volatility or randomness in the pro-

cess  $(\Sigma)$ .

## Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

```
Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

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

LinSDECovY 11

```
sigma <- matrix(
  data = c(
    0.24455556, 0.02201587, -0.05004762,
    0.02201587, 0.07067800, 0.01539456,
    -0.05004762, 0.01539456, 0.07553061
  ),
  nrow = 3
)
LinSDECovEta(
  phi = phi,
    sigma = sigma
)</pre>
```

LinSDECovY

Steady-State Covariance Matrix for the Observed Variables in the Linear Stochastic Differential Equation Model

# Description

The steady-state covariance matrix for the observed variables in the linear stochastic differential equation model Cov(y) is given by

$$\operatorname{Cov}(\mathbf{y}) = \mathbf{\Lambda} \operatorname{Cov}(\boldsymbol{\eta}) \mathbf{\Lambda}' + \mathbf{\Theta}$$

where  $\Lambda$  is the matrix of factor loadings,  $\Theta$  is the covariance matrix of the measurement error, and  $\text{Cov}(\eta)$  is the steady-state covariance matrix for the latent variables.

# Usage

```
LinSDECovY(lambda, theta, cov_eta)
```

# Arguments

lambda	Numeric matrix. Factor loading matrix linking the latent variables to the observed variables ( $\Lambda$ ).
theta	Numeric matrix. The covariance matrix of the measurement error $(\Theta)$ .
cov_eta	Numeric matrix. The steady-state covariance matrix for the latent variables in the linear stochastic differential equation model

## Author(s)

Ivan Jacob Agaloos Pesigan

12 LinSDEMeanEta

## See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()

# **Examples**

```
phi <- matrix(</pre>
  data = c(
    -0.357, 0.771, -0.450,
    0.0, -0.511, 0.729,
    0.0, 0.0, -0.693
  ).
  nrow = 3
)
sigma <- matrix(</pre>
  data = c(
    0.24455556, 0.02201587, -0.05004762,
    0.02201587, 0.07067800, 0.01539456,
    -0.05004762, 0.01539456, 0.07553061
  ),
  nrow = 3
)
lambda <- diag(3)</pre>
theta <- diag(3)
cov_eta <- LinSDECovEta(</pre>
  phi = phi,
  sigma = sigma
LinSDECovY(
  lambda = lambda,
  theta = theta,
  cov_eta = cov_eta
)
```

LinSDEMeanEta

Steady-State Mean Vector for the Latent Variables in the Linear Stochastic Differential Equation Model

# Description

The steady-state mean vector for the latent variables in the linear stochastic differential equation model  $\text{Mean}(\eta)$  is given by

 $\operatorname{Mean}(\boldsymbol{\eta}) = -\boldsymbol{\Phi}^{-1}\boldsymbol{\iota}$ 

where  $\Phi$  is the drift matrix, and  $\iota$  is an unobserved term that is constant over time.

LinSDEMeanY 13

## Usage

```
LinSDEMeanEta(phi, iota)
```

# Arguments

phi Numeric matrix. The drift matrix which represents the rate of change of the

solution in the absence of any random fluctuations  $(\Phi)$ .

iota Numeric vector. An unobserved term that is constant over time  $(\iota)$ .

## Author(s)

Ivan Jacob Agaloos Pesigan

## See Also

```
Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMUINSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

# **Examples**

```
phi <- matrix(
   data = c(
     -0.357, 0.771, -0.450,
     0.0, -0.511, 0.729,
     0.0, 0.0, -0.693
   ),
   nrow = 3
)
iota <- rep(x = 1, times = 3)
LinSDEMeanEta(
   phi = phi,
   iota = iota
)</pre>
```

LinSDEMeanY

Steady-State Mean Vector for the Observed Variables in the Linear Stochastic Differential Equation Model

14 LinSDEMeanY

# **Description**

The steady-state mean vector for the observed variables in the linear stochastic differential equation model Mean  $(\eta)$  is given by

$$\nu + \Lambda \text{Mean}(\eta)$$

where  $\nu$  is the vector of intercept values for the measurement model,  $\Lambda$  is the matrix of factor loadings, and Mean  $(\eta)$  is the steady-state mean vector for the latent variables.

#### Usage

```
LinSDEMeanY(nu, lambda, mean_eta)
```

## **Arguments**

nu Numeric vector. Vector of intercept values for the measurement model  $(\nu)$ .

lambda Numeric matrix. Factor loading matrix linking the latent variables to the ob-

served variables  $(\Lambda)$ .

mean\_eta Numeric vector. Steady-state mean vector of the latent variables Mean  $(\eta)$ .

#### Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

```
Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

```
phi <- matrix(
  data = c(
    -0.357, 0.771, -0.450,
    0.0, -0.511, 0.729,
    0.0, 0.0, -0.693
),
  nrow = 3
)
iota <- rep(x = 1, times = 3)
lambda <- diag(3)
nu <- rep(x = 1, times = 3)
mean_eta <- LinSDEMeanEta(
  phi = phi,
  iota = iota
)
LinSDEMeanY(</pre>
```

plot.simstatespace 15

```
nu = nu,
lambda = lambda,
mean_eta = mean_eta
)
```

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

```
# 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_1 <- t(chol(sigma0))</pre>
```

plot.simstatespace

```
alpha < - rep(x = 0, times = p)
beta <- 0.50 * diag(p)
psi <- diag(p)</pre>
psi_l <- t(chol(psi))</pre>
## measurement model
k <- 3
nu \leftarrow rep(x = 0, times = k)
lambda <- diag(k)</pre>
theta <-0.50 * diag(k)
theta_l <- t(chol(theta))</pre>
## covariates
j <- 2
x \leftarrow lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
  }
)
gamma \leftarrow diag(x = 0.10, nrow = p, ncol = j)
kappa \leftarrow diag(x = 0.10, nrow = k, ncol = j)
# Type 0
ssm <- SimSSMFixed(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_1 = theta_1,
  type = 0
)
plot(ssm)
plot(ssm, id = 1:3, time = 0:9)
# Type 1
ssm <- SimSSMFixed(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
```

print.simstatespace 17

```
lambda = lambda,
  theta_1 = theta_1,
  type = 1,
  x = x,
  gamma = gamma
plot(ssm)
plot(ssm, id = 1:3, time = 0:9)
# Type 2
ssm <- SimSSMFixed(</pre>
  n = n,
  time = time,
 mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_1 = theta_1,
  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.

print.simstatespace

## Value

Prints simulated data in long format.

## Author(s)

Ivan Jacob Agaloos Pesigan

```
# 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)</pre>
sigma0_l <- t(chol(sigma0))</pre>
alpha <- rep(x = 0, times = p)
beta <- 0.50 * diag(p)
psi <- diag(p)</pre>
psi_l <- t(chol(psi))</pre>
## measurement model
k <- 3
nu < -rep(x = 0, times = k)
lambda <- diag(k)</pre>
theta <-0.50 * diag(k)
theta_l <- t(chol(theta))</pre>
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
  }
)
gamma \leftarrow diag(x = 0.10, nrow = p, ncol = j)
kappa \leftarrow diag(x = 0.10, nrow = k, ncol = j)
# Type 0
ssm <- SimSSMFixed(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
```

ProjectToStability 19

```
beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_1 = theta_1,
  type = 0
)
print(ssm)
# Type 1
ssm <- SimSSMFixed(</pre>
  n = n,
  time = time,
 mu0 = mu0,
  sigma0_1 = sigma0_1,
  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(</pre>
 n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  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)
```

20 ProjectToStability

## **Description**

Scales a square matrix so that its spectral radius is strictly less than 1 by a specified stability margin. This is useful for ensuring that transition matrices in state space or vector autoregressive (VAR) models are stationary. If the matrix is already within the margin, it is returned unchanged.

#### Usage

```
ProjectToStability(x, margin = 0.98, tol = 1e-12)
```

# **Arguments**

x Numeric square matrix.

margin Double in (0,1). Target upper bound for the spectral radius (default = 0.98).

tol Small positive double added to the denominator in the scaling factor to avoid division by zero (default = 1e-12).

#### **Details**

The projection is performed by multiplying the matrix by a constant factor  $c = \frac{\text{margin}}{\rho + \text{tol}}$ , where  $\rho$  is the spectral radius and tol is a small positive number to prevent division by zero.

#### Value

A numeric matrix of the same dimensions as x, scaled if necessary to satisfy the stability constraint.

## Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

```
Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUFixed(), SimSSMOUFixed(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

```
# Matrix with eigenvalues greater than 1
x <- matrix(
  data = c(
    1.2, 0.3,
    0.4, 0.9
  ),
   nrow = 2
)
SpectralRadius(x = x) # > 1
SpectralRadius(x = ProjectToStability(x = x)) # < 1</pre>
```

SimAlphaN 21

```
# Matrix already stable is returned unchanged
x <- matrix(
  data = c(
    0.5, 0.3,
    0.2, 0.4
  ),
  nrow = 2
)
identical(ProjectToStability(x = x), x)</pre>
```

SimAlphaN

Simulate Intercept Vectors in a Discrete-Time Vector Autoregressive Model from the Multivariate Normal Distribution

## **Description**

This function simulates random intercept vectors in a discrete-time vector autoregressive model from the multivariate normal distribution.

## Usage

```
SimAlphaN(n, alpha, vcov_alpha_1)
```

## **Arguments**

n Positive integer. Number of replications.

alpha Numeric vector. Intercept  $(\alpha)$ .

vcov\_alpha\_l Numeric matrix. Cholesky factorization (t(chol(vcov\_alpha))) of the sam-

pling variance-covariance matrix of  $\alpha$ .

## Value

Returns a list of random intercept vectors.

## Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

```
Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

22 SimBetaN

## **Examples**

```
n <- 10
alpha <- c(0, 0, 0)
vcov_alpha_l <- t(chol(0.001 * diag(3)))
SimAlphaN(n = n, alpha = alpha, vcov_alpha_l = vcov_alpha_l)</pre>
```

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() with a rejection sampling approach.

## Usage

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

# Arguments

n Positive integer. Number of replications. beta Numeric matrix. The transition matrix  $(\beta)$ . vcov\_beta\_vec\_1

Numeric matrix. Cholesky factorization (t(chol(vcov\_beta\_vec))) of the sampling variance-covariance matrix of  $vec(\beta)$ .

#### Value

Returns a list of random transition matrices.

# Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

```
Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

SimBetaN2 23

# **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)</pre>
```

SimBetaN2

Simulate Transition Matrices from the Multivariate Normal Distribution and Project to Stability

# **Description**

This function simulates random transition matrices from the multivariate normal distribution then projects each draw to the stability region using ProjectToStability().

# Usage

```
SimBetaN2(n, beta, vcov_beta_vec_l, margin = 0.98, tol = 1e-12)
```

## **Arguments**

n Positive integer. Number of replications. beta Numeric matrix. The transition matrix  $(\beta)$ . vcov\_beta\_vec\_1 Numeric matrix. Cholesky factorization (t(chol(vcov\_beta\_vec))) of the sampling variance-covariance matrix of vec  $(\beta)$ . margin Double in (0,1). Target upper bound for the spectral radius (default = 0.98). tol Small positive double added to the denominator in the scaling factor to avoid division by zero (default = 1e-12).

## Value

Returns a list of random transition matrices.

# Author(s)

Ivan Jacob Agaloos Pesigan

24 SimCovDiagN

## See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), 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_1 <- t(chol(0.001 * diag(9)))
SimBetaN2(n = n, beta = beta, vcov_beta_vec_1 = vcov_beta_vec_1)</pre>
```

SimCovDiagN

Simulate Diagonal Covariance Matrices from the Multivariate Normal Distribution

## **Description**

This function simulates random diagonal covariance matrices from the multivariate normal distribution. The function ensures that the generated covariance matrices are positive semi-definite.

# Usage

```
SimCovDiagN(n, sigma_diag, vcov_sigma_diag_l)
```

#### **Arguments**

```
n Positive integer. Number of replications. sigma_diag Numeric matrix. The covariance matrix (\Sigma). vcov_sigma_diag_l
```

Numeric matrix. Cholesky factorization (t(chol(vcov\_sigma\_vech))) of the sampling variance-covariance matrix of vech  $(\Sigma)$ .

## Value

Returns a list of random diagonal covariance matrices.

SimCovN 25

#### Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

```
Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

## **Examples**

```
n <- 10
sigma_diag <- c(1, 1, 1)
vcov_sigma_diag_l <- t(chol(0.001 * diag(3)))
SimCovDiagN(
    n = n,
    sigma_diag = sigma_diag,
    vcov_sigma_diag_l = vcov_sigma_diag_l
)</pre>
```

SimCovN

Simulate Covariance Matrices from the Multivariate Normal Distribution

# **Description**

This function simulates random covariance matrices from the multivariate normal distribution. The function ensures that the generated covariance matrices are positive semi-definite.

# Usage

```
SimCovN(n, sigma, vcov_sigma_vech_l)
```

## **Arguments**

```
n Positive integer. Number of replications. sigma Numeric matrix. The covariance matrix (\Sigma). vcov_sigma_vech_1 Numeric matrix. Cholesky factorization (t(chol(vcov_sigma_vech))) of the sampling variance-covariance matrix of vech (\Sigma).
```

## Value

Returns a list of random covariance matrices.

26 SimIotaN

#### Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN(), SimCovDiagN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()

## **Examples**

```
n <- 10
sigma <- matrix(</pre>
  data = c(
    1.0, 0.5, 0.3,
    0.5, 1.0, 0.4,
    0.3, 0.4, 1.0
  ),
  nrow = 3
)
vcov_sigma_vech_l <- t(</pre>
  chol(
    0.001 * diag(3 * (3 + 1) / 2)
SimCovN(
  n = n,
  sigma = sigma,
  vcov_sigma_vech_l = vcov_sigma_vech_l
)
```

SimIotaN

Simulate Intercept Vectors in a Continuous-Time Vector Autoregressive Model from the Multivariate Normal Distribution

# **Description**

This function simulates random intercept vectors in a continuous-time vector autoregressive model from the multivariate normal distribution.

## Usage

```
SimIotaN(n, iota, vcov_iota_l)
```

SimPhiN 27

## **Arguments**

n Positive integer. Number of replications.

iota Numeric vector. Intercept  $(\iota)$ .

vcov\_iota\_1 Numeric matrix. Cholesky factorization (t(chol(vcov\_iota))) of the sam-

pling variance-covariance matrix of  $\iota$ .

#### Value

Returns a list of random intercept vectors.

## Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

```
Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

## **Examples**

```
n <- 10
iota <- c(0, 0, 0)
vcov_iota_l <- t(chol(0.001 * diag(3)))
SimIotaN(n = n, iota = iota, vcov_iota_l = vcov_iota_l)</pre>
```

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)
```

28 SimSSMFixed

## **Arguments**

```
n Positive integer. Number of replications. 
phi Numeric matrix. The drift matrix (\Phi). 
vcov_phi_vec_1 Numeric matrix. Cholesky factorization (t(chol(vcov_phi_vec))) of the sampling variance-covariance matrix of vec (\Phi).
```

## Value

Returns a list of random drift matrices.

## Author(s)

Ivan Jacob Agaloos Pesigan

## See Also

```
Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), 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_1 <- t(chol(0.001 * diag(9)))
SimPhiN(n = n, phi = phi, vcov_phi_vec_1 = vcov_phi_vec_1)</pre>
```

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.

SimSSMFixed 29

# Usage

```
SimSSMFixed(
 n,
  time,
 delta_t = 1,
 mu0,
 sigma0_l,
 alpha,
 beta,
 psi_l,
 nu,
  lambda,
  theta_1,
  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 $(\mu_{\eta 0})$ .
sigma0_l	Numeric matrix. Cholesky factorization (t(chol(sigma0))) of the covariance matrix of initial latent variable values ( $\Sigma_{\eta 0}$ ).
alpha	Numeric vector. Vector of constant values for the dynamic model $(\alpha)$ .
beta	Numeric matrix. Transition matrix relating the values of the latent variables at the previous to the current time point $(\beta)$ .
psi_l	Numeric matrix. Cholesky factorization (t(chol(psi))) of the covariance matrix of the process noise ( $\Psi$ ).
nu	Numeric vector. Vector of intercept values for the measurement model $(\nu)$ .
lambda	Numeric matrix. Factor loading matrix linking the latent variables to the observed variables $(\Lambda)$ .
theta_l	Numeric matrix. Cholesky factorization ( $t(chol(theta))$ ) of the covariance matrix of the measurement error ( $\Theta$ ).
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 $(\Gamma)$ .
kappa	Numeric matrix. Matrix linking the covariates to the observed variables at current time point $(\kappa)$ .

#### **Details**

#### Type 0:

The measurement model is given by

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

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

$$oldsymbol{arepsilon}_{i,t} = oldsymbol{\Theta}^{rac{1}{2}} \mathbf{z}_{i,t}, \quad ext{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}
ight)$$

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

$$oldsymbol{\eta}_{i,t} = oldsymbol{lpha} + oldsymbol{eta} oldsymbol{\eta}_{i,t-1} + oldsymbol{\zeta}_{i,t}, \quad ext{with} \quad oldsymbol{\zeta}_{i,t} \sim \mathcal{N}\left( oldsymbol{0}, oldsymbol{\Psi} 
ight)$$

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

An alternative representation of the dynamic noise is given by

$$oldsymbol{\zeta}_{i,t} = oldsymbol{\Psi}^{rac{1}{2}} oldsymbol{\mathbf{z}}_{i,t}, \quad ext{with} \quad oldsymbol{\mathbf{z}}_{i,t} \sim \mathcal{N}\left( oldsymbol{0}, \mathbf{I} 
ight)$$

where 
$$\left( \mathbf{\Psi}^{rac{1}{2}} 
ight) \left( \mathbf{\Psi}^{rac{1}{2}} 
ight)' = \mathbf{\Psi}.$$

#### Type 1:

The measurement model is given by

$$\mathbf{y}_{i,t} = \boldsymbol{\nu} + \boldsymbol{\Lambda} \boldsymbol{\eta}_{i,t} + \boldsymbol{arepsilon}_{i,t}, \quad ext{with} \quad \boldsymbol{arepsilon}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, oldsymbol{\Theta}
ight).$$

The dynamic structure is given by

$$oldsymbol{\eta}_{i,t} = oldsymbol{lpha} + oldsymbol{eta} oldsymbol{\eta}_{i,t-1} + oldsymbol{\Gamma} \mathbf{x}_{i,t} + oldsymbol{\zeta}_{i,t}, \quad ext{with} \quad oldsymbol{\zeta}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, oldsymbol{\Psi}
ight)$$

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

## Type 2:

The measurement model is given by

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

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

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

SimSSMFixed 31

#### 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 1, where 1 is the value of the function argument time.
  - time: A vector time points of length 1.
  - y: A 1 by k matrix of values for the manifest variables.
  - eta: A 1 by p matrix of values for the latent variables.
  - x: A 1 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(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()

```
# 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_1 <- t(chol(sigma0))
alpha <- rep(x = 0, times = p)</pre>
```

32 SimSSMFixed

```
beta <- 0.50 * diag(p)
psi <- 0.001 * diag(p)
psi_l <- t(chol(psi))</pre>
## measurement model
k <- 3
nu \leftarrow rep(x = 0, times = k)
lambda <- diag(k)</pre>
theta <- 0.001 * diag(k)
theta_l <- t(chol(theta))</pre>
## covariates
j <- 2
x <- lapply(
  X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
  }
)
gamma \leftarrow diag(x = 0.10, nrow = p, ncol = j)
kappa \leftarrow diag(x = 0.10, nrow = k, ncol = j)
# Type 0
ssm <- SimSSMFixed(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)
plot(ssm)
# Type 1
ssm <- SimSSMFixed(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_1 = psi_1,
  nu = nu,
  lambda = lambda,
  theta_1 = theta_1,
```

```
type = 1,
  x = x,
  gamma = gamma
)
plot(ssm)
# Type 2
ssm <- SimSSMFixed(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  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 $(\mu_{\eta 0})$ .
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 $(\Sigma_{\eta 0})$ .
alpha	List of numeric vectors. Each element of the list is the vector of constant values for the dynamic model $(\alpha)$ .
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 $(\beta)$ .
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 $(\Psi)$ .
nu	List of numeric vectors. Each element of the list is the vector of intercept values for the measurement model $(\nu)$ .
lambda	List of numeric matrices. Each element of the list is the factor loading matrix linking the latent variables to the observed variables $(\Lambda)$ .
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 $(\Theta)$ .
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 $(\Gamma)$ .
kappa	List of numeric matrices. Each element of the list is the matrix linking the covariates to the observed variables at current time point $(\kappa)$ .

# **Details**

Parameters can vary across individuals by providing a list of parameter values. If the length of any of the parameters (mu0,  $sigma0_1$ , alpha, beta,  $psi_1$ , nu, lambda, theta\_1, 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 1, where 1 is the value of the function argument time.
  - time: A vector time points of length 1.
  - y: A 1 by k matrix of values for the manifest variables.
  - eta: A 1 by p matrix of values for the latent variables.
  - x: A 1 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(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()

```
# 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)
)</pre>
```

```
sigma0 <- 0.001 * diag(p)
sigma0_1 \leftarrow list(
  t(chol(sigma0))
)
alpha <- list(</pre>
  rep(x = 0, times = p)
beta <- list(</pre>
  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(</pre>
  t(chol(psi))
## measurement model
k <- 3
nu <- list(
  rep(x = 0, times = k)
lambda <- list(</pre>
  diag(k)
theta <- 0.001 * diag(k)
theta_l <- list(</pre>
  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(</pre>
  diag(x = 0.10, nrow = p, ncol = j)
kappa <- list(</pre>
  diag(x = 0.10, nrow = k, ncol = j)
# Type 0
ssm <- SimSSMIVary(</pre>
  n = n,
  time = time,
```

```
mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)
plot(ssm)
# Type 1
ssm <- SimSSMIVary(</pre>
 n = n,
  time = time,
 mu0 = mu0,
  sigma0_1 = sigma0_1,
  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(</pre>
 n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  nu = nu,
  lambda = lambda,
  theta_1 = theta_1,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
)
plot(ssm)
```

 ${\tt SimSSMLinGrowth}$ 

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 $(\Gamma)$ .
kappa	Numeric matrix. Matrix linking the covariates to the observed variables at current time point $(\kappa)$ .

## **Details**

# Type 0:

The measurement model is given by

$$Y_{i,t} = \left( egin{array}{cc} 1 & 0 \end{array} 
ight) \left( egin{array}{c} \eta_{0_{i,t}} \\ \eta_{1_{i,t}} \end{array} 
ight) + oldsymbol{arepsilon}_{i,t}, \quad ext{with} \quad oldsymbol{arepsilon}_{i,t} \sim \mathcal{N}\left(0, heta
ight)$$

where  $Y_{i,t}$ ,  $\eta_{0_{i,t}}$ ,  $\eta_{1_{i,t}}$ , and  $\varepsilon_{i,t}$  are random variables and  $\theta$  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  $\varepsilon_{i,t}$  a vector of random measurement errors at time t and individual i.  $\theta$  is the variance of  $\varepsilon$ .

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

$$oldsymbol{\mu_{\eta|0}} = \left(egin{array}{c} \mu_{\eta_0} \ \mu_{\eta_1} \end{array}
ight) \quad ext{and},$$

$$oldsymbol{\Sigma_{\eta|0}} = \left(egin{array}{cc} \sigma_{\eta_0}^2 & \sigma_{\eta_0,\eta_1} \ \sigma_{\eta_1,\eta_0} & \sigma_{\eta_1}^2 \end{array}
ight).$$

## Type 1:

The measurement model is given by

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

The dynamic structure is given by

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

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

### Type 2:

The measurement model is given by

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

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

The dynamic structure is given by

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

### 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 1, where 1 is the value of the function argument time.
- time: A vector time points of length 1.
- y: A 1 by k matrix of values for the manifest variables.
- eta: A 1 by p matrix of values for the latent variables.
- x: A 1 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(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()

```
# 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(</pre>
  data = c(
    1.932,
    0.618,
    0.618,
    0.587
  ),
  nrow = p
sigma0_1 \leftarrow t(chol(sigma0))
```

```
## measurement model
k <- 1
theta <- 0.50
theta_l <- sqrt(theta)</pre>
## covariates
j <- 2
x <- lapply(
 X = seq_len(n),
 FUN = function(i) {
    matrix(
      data = rnorm(n = j * time),
      nrow = j
 }
)
gamma \leftarrow diag(x = 0.10, nrow = p, ncol = j)
kappa <- diag(x = 0.10, nrow = k, ncol = j)
# Type 0
ssm <- SimSSMLinGrowth(</pre>
 n = n,
 time = time,
 mu0 = mu0,
  sigma0_1 = sigma0_1,
  theta_l = theta_l,
  type = 0
)
plot(ssm)
# Type 1
ssm <- SimSSMLinGrowth(</pre>
 n = n,
 time = time,
 mu0 = mu0,
  sigma0_1 = sigma0_1,
  theta_1 = theta_1,
  type = 1,
  x = x,
  gamma = gamma
plot(ssm)
# Type 2
ssm <- SimSSMLinGrowth(</pre>
  n = n,
  time = time,
 mu0 = mu0,
  sigma0_1 = sigma0_1,
  theta_l = theta_l,
  type = 2,
  x = x,
```

```
gamma = gamma,
kappa = kappa
)
plot(ssm)
```

 ${\tt SimSSMLinGrowthIVary}$ 

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 ${\tt SimSSMLinGrowth()}$ 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 $(\Gamma)$ .
kappa	List of numeric matrices. Each element of the list is the matrix linking the
	covariates to the observed variables at current time point $(\kappa)$ .

#### **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\_1, 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 1, where 1 is the value of the function argument time.
  - time: A vector time points of length 1.
  - y: A 1 by k matrix of values for the manifest variables.
  - eta: A 1 by p matrix of values for the latent variables.
  - x: A 1 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(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(),

```
SSMMeanY(), SimAlphaN(), SimBetaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(),
SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(),
SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(),
TestPhi(), TestStability(), TestStationarity()
```

```
# 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 \leftarrow 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)</pre>
sigma0 <- matrix(</pre>
  data = c(
    1.932,
    0.618,
    0.618,
    0.587
  ),
  nrow = p
)
sigma0_l <- list(t(chol(sigma0)))</pre>
## measurement model
k <- 1
theta <- 0.50
theta_l <- list(sqrt(theta))</pre>
## covariates
j <- 2
x \leftarrow lapply(
 X = seq_len(n),
 FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
 }
gamma <- list(</pre>
 diag(x = 0.10, nrow = p, ncol = j)
kappa <- list(</pre>
  diag(x = 0.10, nrow = k, ncol = j)
```

```
)
# Type 0
ssm <- SimSSMLinGrowthIVary(</pre>
 n = n,
 time = time,
 mu0 = mu0,
  sigma0_1 = sigma0_1,
  theta_l = theta_l,
  type = 0
)
plot(ssm)
# Type 1
ssm <- SimSSMLinGrowthIVary(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  theta_l = theta_l,
  type = 1,
  x = x,
  gamma = gamma
plot(ssm)
# Type 2
ssm <- SimSSMLinGrowthIVary(</pre>
 n = n,
 time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  theta_1 = theta_1,
  type = 2,
  x = x,
  gamma = gamma,
  kappa = kappa
plot(ssm)
```

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_1,
  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 $(\Delta_t)$ .
mu0	Numeric vector. Mean of initial latent variable values $(\mu_{\eta 0})$ .
sigma0_l	Numeric matrix. Cholesky factorization (t(chol(sigma0))) of the covariance matrix of initial latent variable values $(\Sigma_{\eta 0})$ .
iota	Numeric vector. An unobserved term that is constant over time $(\iota)$ .
phi	Numeric matrix. The drift matrix which represents the rate of change of the solution in the absence of any random fluctuations $(\Phi)$ .
sigma_l	Numeric matrix. Cholesky factorization (t(chol(sigma))) of the covariance matrix of volatility or randomness in the process $(\Sigma)$ .
nu	Numeric vector. Vector of intercept values for the measurement model $(\nu)$ .
lambda	Numeric matrix. Factor loading matrix linking the latent variables to the observed variables $(\Lambda)$ .
theta_l	Numeric matrix. Cholesky factorization (t(chol(theta))) of the covariance matrix of the measurement error $(\Theta)$ .
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. Rumeric matrix. Matrix linking the covariates to the latent variables at current time point  $(\Gamma)$ .

Rappa Numeric matrix. Matrix linking the covariates to the observed variables at current time point  $(\kappa)$ .

#### **Details**

## Type 0:

The measurement model is given by

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

where  $\mathbf{y}_{i,t}$ ,  $\eta_{i,t}$ , and  $\varepsilon_{i,t}$  are random variables and  $\nu$ ,  $\Lambda$ , and  $\Theta$  are model parameters.  $\mathbf{y}_{i,t}$  represents a vector of observed random variables,  $\eta_{i,t}$  a vector of latent random variables, and  $\varepsilon_{i,t}$  a vector of random measurement errors, at time t and individual t.  $\nu$  denotes a vector of intercepts,  $\Lambda$  a matrix of factor loadings, and  $\Theta$  the covariance matrix of  $\varepsilon$ .

An alternative representation of the measurement error is given by

$$oldsymbol{arepsilon}_{i,t} = oldsymbol{\Theta}^{rac{1}{2}} \mathbf{z}_{i,t}, \quad ext{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}\right)$$

where  $\mathbf{z}_{i,t}$  is a vector of independent standard normal random variables and  $\left(\Theta^{\frac{1}{2}}\right)\left(\Theta^{\frac{1}{2}}\right)' = \Theta$ . 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  $\iota$  is a term which is unobserved and constant over time,  $\Phi$  is the drift matrix which represents the rate of change of the solution in the absence of any random fluctuations,  $\Sigma$  is the matrix of volatility or randomness in the process, and  $\mathrm{d}W$  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

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

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

## **Type 2:**

The measurement model is given by

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

where  $\kappa$  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

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

$$oldsymbol{lpha}_{\Delta t_{l_{s}}} = oldsymbol{\Phi}^{-1} \left(oldsymbol{eta} - \mathbf{I}_{p}
ight) oldsymbol{\iota}$$

$$\operatorname{vec}\left(\mathbf{\Psi}_{\Delta t_{l_{i}}}\right) = \left[\left(\mathbf{\Phi} \otimes \mathbf{I}_{p}\right) + \left(\mathbf{I}_{p} \otimes \mathbf{\Phi}\right)\right] \left[\exp\left(\left[\left(\mathbf{\Phi} \otimes \mathbf{I}_{p}\right) + \left(\mathbf{I}_{p} \otimes \mathbf{\Phi}\right)\right] \Delta t\right) - \mathbf{I}_{p \times p}\right] \operatorname{vec}\left(\mathbf{\Sigma}\right)$$

where p is the number of latent variables and  $\Delta t$  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 1, where 1 is the value of the function argument time.
  - time: A vector time points of length 1.
  - y: A 1 by k matrix of values for the manifest variables.
  - eta: A 1 by p matrix of values for the latent variables.
  - x: A 1 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(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()

```
# 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))</pre>
iota <- c(0.317, 0.230)
phi <- matrix(</pre>
  data = c(
    -0.10,
    0.05,
    0.05,
    -0.10
  ),
  nrow = p
)
sigma <- matrix(</pre>
  data = c(
    2.79,
    0.06,
    0.06,
    3.27
  ),
  nrow = p
)
sigma_l <- t(chol(sigma))</pre>
## measurement model
k <- 2
nu \leftarrow rep(x = 0, times = k)
lambda <- diag(k)</pre>
theta <-0.001 * diag(k)
theta_l <- t(chol(theta))</pre>
## 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 \leftarrow diag(x = 0.10, nrow = k, ncol = j)
# Type 0
ssm <- SimSSMLinSDEFixed(</pre>
 n = n,
  time = time,
  delta_t = delta_t,
 mu0 = mu0,
  sigma0_1 = sigma0_1,
  iota = iota,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_1 = theta_1,
  type = 0
)
plot(ssm)
# Type 1
ssm <- SimSSMLinSDEFixed(</pre>
 n = n,
 time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  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(</pre>
  n = n,
  time = time,
```

SimSSMLinSDEIVary 51

```
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(
  time,
  delta_t = 1,
 mu0,
  sigma0_1,
  iota,
  phi,
  sigma_l,
  nu,
  lambda,
  theta_1,
  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 $(\mu_{\eta 0})$ .
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 $(\Sigma_{\eta 0})$ .
iota	List of numeric vectors. Each element of the list is an unobserved term that is constant over time $(\iota)$ .
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 $(\Phi)$ .
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 $\Sigma$ .
nu	List of numeric vectors. Each element of the list is the vector of intercept values for the measurement model $(\nu)$ .
lambda	List of numeric matrices. Each element of the list is the factor loading matrix linking the latent variables to the observed variables ( $\Lambda$ ).
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 $(\Theta)$ .
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 $(\Gamma)$ .
kappa	List of numeric matrices. Each element of the list is the matrix linking the covariates to the observed variables at current time point $(\kappa)$ .

## **Details**

Parameters can vary across individuals by providing a list of parameter values. If the length of any of the parameters  $(mu0, sigma0_1, iota, phi, sigma_1, nu, lambda, theta_1, gamma, or kappa)$  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 1, where 1 is the value of the function argument time

53

- time: A vector time points of length 1.
- y: A 1 by k matrix of values for the manifest variables.
- eta: A 1 by p matrix of values for the latent variables.
- x: A 1 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(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

```
# 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(</pre>
```

```
c(-3.0, 1.5)
)
sigma0 <- 0.001 * diag(p)
sigma0_l <- list(</pre>
  t(chol(sigma0))
)
iota <- list(</pre>
  c(0.317, 0.230)
)
phi <- list(</pre>
 -0.1 * diag(p),
  -0.2 * diag(p),
  -0.3 * diag(p),
  -0.4 * diag(p),
  -0.5 * diag(p)
)
sigma <- matrix(</pre>
  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(</pre>
  diag(k)
)
theta <- 0.001 * diag(k)
theta_l <- list(</pre>
  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(</pre>
```

SimSSMLinSDEIVary

```
diag(x = 0.10, nrow = p, ncol = j)
kappa <- list(</pre>
 diag(x = 0.10, nrow = k, ncol = j)
)
# Type 0
ssm <- SimSSMLinSDEIVary(</pre>
 n = n,
 time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  iota = iota,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)
plot(ssm)
# Type 1
ssm <- SimSSMLinSDEIVary(</pre>
 n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  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(</pre>
 n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  iota = iota,
  phi = phi,
```

```
sigma_1 = sigma_1,
nu = nu,
lambda = lambda,
theta_1 = theta_1,
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(
  time,
  delta_t = 1,
 mu0,
  sigma0_1,
 mu,
  phi,
  sigma_l,
  nu,
  lambda,
  theta_1,
  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  $(\Delta_t)$ .

mu0	Numeric vector. Mean of initial latent variable values $(\mu_{\eta 0})$ .
sigma0_l	Numeric matrix. Cholesky factorization (t(chol(sigma0))) of the covariance matrix of initial latent variable values $(\Sigma_{\eta 0})$ .
mu	Numeric vector. The long-term mean or equilibrium level $(\mu)$ .
phi	Numeric matrix. The drift matrix which represents the rate of change of the solution in the absence of any random fluctuations ( $\Phi$ ). 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 $(\Sigma)$ .
nu	Numeric vector. Vector of intercept values for the measurement model $(\nu)$ .
lambda	Numeric matrix. Factor loading matrix linking the latent variables to the observed variables ( $\Lambda$ ).
theta_l	Numeric matrix. Cholesky factorization (t(chol(theta))) of the covariance matrix of the measurement error $(\Theta)$ .
type	Integer. State space model type. See Details for more information.
Х	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 ( $\Gamma$ ).
kappa	Numeric matrix. Matrix linking the covariates to the observed variables at current time point $(\kappa)$ .

### **Details**

## Type 0:

The measurement model is given by

$$\mathbf{y}_{i,t} = \mathbf{\nu} + \mathbf{\Lambda} oldsymbol{\eta}_{i,t} + oldsymbol{arepsilon}_{i,t}, \quad ext{with} \quad oldsymbol{arepsilon}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{\Theta}
ight)$$

where  $\mathbf{y}_{i,t}$ ,  $\eta_{i,t}$ , and  $\varepsilon_{i,t}$  are random variables and  $\nu$ ,  $\Lambda$ , and  $\Theta$  are model parameters.  $\mathbf{y}_{i,t}$  represents a vector of observed random variables,  $\eta_{i,t}$  a vector of latent random variables, and  $\varepsilon_{i,t}$  a vector of random measurement errors, at time t and individual i.  $\nu$  denotes a vector of intercepts,  $\Lambda$  a matrix of factor loadings, and  $\Theta$  the covariance matrix of  $\varepsilon$ .

An alternative representation of the measurement error is given by

$$oldsymbol{arepsilon}_{i,t} = oldsymbol{\Theta}^{rac{1}{2}} \mathbf{z}_{i,t}, \quad ext{with} \quad \mathbf{z}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{I}
ight)$$

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

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

where  $\mu$  is the long-term mean or equilibrium level,  $\Phi$  is the rate of mean reversion, determining how quickly the variable returns to its mean,  $\Sigma$  is the matrix of volatility or randomness in the process, and dW 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

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

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

## Type 2:

The measurement model is given by

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

where  $\kappa$  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

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

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

## 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 1, where 1 is the value of the function argument time.
  - time: A vector time points of length 1.
  - y: A 1 by k matrix of values for the manifest variables.
  - eta: A 1 by p matrix of values for the latent variables.
  - x: A 1 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(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()

```
# 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))</pre>
mu < -c(5.76, 5.18)
phi <- matrix(</pre>
  data = c(
    -0.10,
    0.05,
    0.05,
    -0.10
  ),
  nrow = p
sigma <- matrix(</pre>
```

```
data = c(
    2.79,
    0.06,
    0.06,
    3.27
  ),
  nrow = p
)
sigma_l <- t(chol(sigma))</pre>
## measurement model
k <- 2
nu \leftarrow rep(x = 0, times = k)
lambda <- diag(k)</pre>
theta <- 0.001 * diag(k)
theta_l \leftarrow 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 \leftarrow diag(x = 0.10, nrow = p, ncol = j)
kappa <- diag(x = 0.10, nrow = k, ncol = j)
# Type 0
ssm <- SimSSMOUFixed(</pre>
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  mu = mu,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_l = theta_l,
  type = 0
)
plot(ssm)
# Type 1
ssm <- SimSSMOUFixed(</pre>
  n = n,
  time = time,
  delta_t = delta_t,
```

```
mu0 = mu0,
  sigma0_1 = sigma0_1,
  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(</pre>
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  mu = mu,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_1 = theta_1,
  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.

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  $(\mu_{\eta|0})$ .

sigma0\_1 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

 $(\Sigma_{\eta|0}).$ 

mu List of numeric vectors. Each element of the list is the long-term mean or equi-

librium level ( $\mu$ ).

phi List of numeric matrix. Each element of the list is the drift matrix which repre-

sents the rate of change of the solution in the absence of any random fluctuations  $(\Phi)$ . It also represents the rate of mean reversion, determining how quickly the

variable returns to its mean.

sigma\_1 List of numeric matrix. Each element of the list is the Cholesky factorization

 $(t(\mbox{chol}(\mbox{sigma})))$  of the covariance matrix of volatility or randomness in the

process  $\Sigma$ .

nu List of numeric vectors. Each element of the list is the vector of intercept values

for the measurement model ( $\nu$ ).

lambda List of numeric matrices. Each element of the list is the factor loading matrix

linking the latent variables to the observed variables ( $\Lambda$ ).

theta\_1 List of numeric matrices. Each element of the list is the Cholesky factorization

(t(chol(theta))) of the covariance matrix of the measurement error  $(\Theta)$ .

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 $(\Gamma)$ .
kappa	List of numeric matrices. Each element of the list is the matrix linking the
	covariates to the observed variables at current time point $(\kappa)$

#### **Details**

Parameters can vary across individuals by providing a list of parameter values. If the length of any of the parameters (mu0, sigma0\_1, mu, phi, sigma\_1, nu, lambda, theta\_1, 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 1, where 1 is the value of the function argument time.
  - time: A vector time points of length 1.
  - y: A 1 by k matrix of values for the manifest variables.
  - eta: A 1 by p matrix of values for the latent variables.
  - x: A 1 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(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()

```
# 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 \leftarrow list(
  t(chol(sigma0))
)
mu <- list(</pre>
  c(5.76, 5.18)
phi <- list(</pre>
  -0.1 * diag(p),
  -0.2 * diag(p),
  -0.3 * diag(p),
  -0.4 * diag(p),
  -0.5 * diag(p)
)
sigma <- matrix(</pre>
  data = c(
    2.79,
    0.06,
    0.06,
    3.27
  ),
  nrow = p
sigma_l <- list(</pre>
  t(chol(sigma))
## measurement model
k <- 2
nu <- list(</pre>
```

```
rep(x = 0, times = k)
lambda <- list(</pre>
  diag(k)
)
theta <- 0.001 * diag(k)
theta_l <- list(</pre>
 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(</pre>
 diag(x = 0.10, nrow = p, ncol = j)
kappa <- list(</pre>
  diag(x = 0.10, nrow = k, ncol = j)
# Type 0
ssm <- SimSSMOUIVary(</pre>
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  mu = mu,
  phi = phi,
  sigma_l = sigma_l,
  nu = nu,
  lambda = lambda,
  theta_1 = theta_1,
  type = 0
)
plot(ssm)
# Type 1
ssm <- SimSSMOUIVary(</pre>
  n = n,
  time = time,
  delta_t = delta_t,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
```

```
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(</pre>
 n = n,
 time = time,
 delta_t = delta_t,
 mu0 = mu0,
 sigma0_1 = sigma0_1,
 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_1,
alpha,
beta,
psi_1,
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 $(\mu_{\eta 0})$ .
sigma0_l	Numeric matrix. Cholesky factorization (t(chol(sigma0))) of the covariance matrix of initial latent variable values ( $\Sigma_{\eta 0}$ ).
alpha	Numeric vector. Vector of constant values for the dynamic model $(\alpha)$ .
beta	Numeric matrix. Transition matrix relating the values of the latent variables at the previous to the current time point $(\beta)$ .
psi_l	Numeric matrix. Cholesky factorization (t(chol(psi))) of the covariance matrix of the process noise ( $\Psi$ ).
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 $(\Gamma)$ .

## **Details**

### Type 0:

The measurement model is given by

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

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

$$oldsymbol{\eta}_{i,t} = oldsymbol{lpha} + oldsymbol{eta} oldsymbol{\eta}_{i,t-1} + oldsymbol{\zeta}_{i,t}, \quad ext{with} \quad oldsymbol{\zeta}_{i,t} \sim \mathcal{N}\left(oldsymbol{0}, oldsymbol{\Psi}
ight)$$

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

An alternative representation of the dynamic noise is given by

$$oldsymbol{\zeta}_{i,t} = oldsymbol{\Psi}^{rac{1}{2}} oldsymbol{\mathbf{z}}_{i,t}, \quad ext{with} \quad oldsymbol{\mathbf{z}}_{i,t} \sim \mathcal{N}\left(oldsymbol{0}, oldsymbol{\mathbf{I}}
ight)$$

where 
$$\left(\Psi^{rac{1}{2}}
ight)\left(\Psi^{rac{1}{2}}
ight)'=\Psi.$$

## Type 1:

The measurement model is given by

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

The dynamic structure is given by

$$oldsymbol{\eta}_{i,t} = oldsymbol{lpha} + oldsymbol{eta} oldsymbol{\eta}_{i,t-1} + oldsymbol{\Gamma} \mathbf{x}_{i,t} + oldsymbol{\zeta}_{i,t}, \quad ext{with} \quad oldsymbol{\zeta}_{i,t} \sim \mathcal{N}\left(\mathbf{0}, oldsymbol{\Psi}
ight)$$

where  $\mathbf{x}_{i,t}$  represents a vector of covariates at time t and individual i, and  $\Gamma$  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 1, where 1 is the value of the function argument time.
  - time: A vector time points of length 1.
  - y: A 1 by k matrix of values for the manifest variables.
  - eta: A 1 by p matrix of values for the latent variables.
  - x: A 1 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(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()

```
# 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))</pre>
alpha <- rep(x = 0, times = p)
beta <- 0.50 * diag(p)
psi <- 0.001 * diag(p)
psi_l <- t(chol(psi))</pre>
## covariates
j <- 2
x \leftarrow lapply(
 X = seq_len(n),
  FUN = function(i) {
    matrix(
      data = stats::rnorm(n = time * j),
      nrow = j,
      ncol = time
    )
 }
gamma \leftarrow diag(x = 0.10, nrow = p, ncol = j)
# Type 0
ssm <- SimSSMVARFixed(</pre>
 n = n,
  time = time,
 mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  type = 0
)
plot(ssm)
# Type 1
ssm <- SimSSMVARFixed(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
```

70 SimSSMVARIVary

```
psi_l = psi_l,
  type = 1,
  x = x,
  gamma = gamma
)
plot(ssm)
```

 ${\tt SimSSMVARIVary}$ 

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 $(\mu_{\eta 0})$ .
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 $(\Sigma_{\eta 0})$ .
alpha	List of numeric vectors. Each element of the list is the vector of constant values for the dynamic model $(\alpha)$ .
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 $(\beta)$ .

SimSSMVARIVary 71

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 $(\Psi)$ .
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 $(\Gamma)$ .

## **Details**

Parameters can vary across individuals by providing a list of parameter values. If the length of any of the parameters (mu0, sigma0\_1, alpha, beta, psi\_1, 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 1, where 1 is the value of the function argument time.
  - time: A vector time points of length 1.
  - y: A 1 by k matrix of values for the manifest variables.
  - eta: A 1 by p matrix of values for the latent variables.
  - x: A 1 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(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

72 SimSSMVARIVary

```
# 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(</pre>
  t(chol(sigma0))
alpha <- list(</pre>
  rep(x = 0, times = p)
beta <- list(</pre>
  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)</pre>
psi_l <- list(</pre>
  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(</pre>
  diag(x = 0.10, nrow = p, ncol = j)
# Type 0
ssm <- SimSSMVARIVary(</pre>
 n = n,
 time = time,
  mu0 = mu0,
```

SpectralRadius 73

```
sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  type = 0
)
plot(ssm)
# Type 1
ssm <- SimSSMVARIVary(</pre>
  n = n,
  time = time,
  mu0 = mu0,
  sigma0_1 = sigma0_1,
  alpha = alpha,
  beta = beta,
  psi_l = psi_l,
  type = 1,
  x = x,
  gamma = gamma
plot(ssm)
```

SpectralRadius

Spectral Radius

# Description

Computes the spectral radius of a square matrix, defined as the maximum modulus (absolute value) of its eigenvalues. The spectral radius is often used to assess the stability of systems such as vector autoregressive (VAR) models: a system is considered stationary if the spectral radius of its transition matrix is strictly less than 1.

## Usage

```
SpectralRadius(x)
```

# Arguments

Х

Numeric square matrix.

#### Value

Numeric value representing the spectral radius of x.

74 SSMCovEta

#### Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), TestPhi(), TestStability(), TestStationarity()

#### **Examples**

```
# Matrix with eigenvalues less than 1
x <- matrix(</pre>
  data = c(
    0.5, 0.3,
    0.2, 0.4
  ),
  nrow = 2
SpectralRadius(x)
# Matrix with eigenvalues greater than 1
y <- matrix(</pre>
  data = c(
    1.2, 0.3,
    0.4, 0.9
  ),
  nrow = 2
SpectralRadius(y)
```

SSMCovEta

Steady-State Covariance Matrix for the Latent Variables in the State Space Model

#### **Description**

The steady-state covariance matrix for the latent variables in the state space model  $\mathrm{Cov}\left(\eta\right)$  is given by

$$\operatorname{vec}\left(\operatorname{Cov}\left(\boldsymbol{\eta}\right)\right) = \left(\mathbf{I} - \boldsymbol{\beta} \otimes \boldsymbol{\beta}\right)^{-1} \operatorname{vec}\left(\boldsymbol{\Psi}\right)$$

where  $\beta$  is the transition matrix relating the values of the latent variables at the previous to the current time point and  $\Psi$  is the covariance matrix of volatility or randomness in the process.

SSMCovY 75

#### Usage

```
SSMCovEta(beta, psi)
```

# Arguments

beta Numeric matrix. Transition matrix relating the values of the latent variables at

the previous to the current time point  $(\beta)$ .

Numeric matrix. The covariance matrix of the process noise  $(\Psi)$ .

#### Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

```
Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

## **Examples**

```
beta <- matrix(
  data = c(
    0.7, 0.5, -0.1,
    0.0, 0.6, 0.4,
    0.0, 0.0, 0.5
  ),
  nrow = 3
)
psi <- 0.1 * diag(3)
SSMCovEta(
  beta = beta,
  psi = psi
)</pre>
```

SSMCovY

Steady-State Covariance Matrix for the Observed Variables in the State Space Model

76 SSMCovY

# **Description**

The steady-state covariance matrix for the observed variables in the state space model Cov(y) is given by

$$Cov(\mathbf{y}) = \mathbf{\Lambda}Cov(\mathbf{\eta})\mathbf{\Lambda}' + \mathbf{\Theta}$$

where  $\Lambda$  is the matrix of factor loadings,  $\Theta$  is the covariance matrix of the measurement error, and  $\text{Cov}(\eta)$  is the steady-state covariance matrix for the latent variables.

#### Usage

```
SSMCovY(lambda, theta, cov_eta)
```

## **Arguments**

lambda Numeric matrix. Factor loading matrix linking the latent variables to the ob-

served variables ( $\Lambda$ ).

theta Numeric matrix. The covariance matrix of the measurement error  $(\Theta)$ .

cov\_eta Numeric matrix. The steady-state covariance matrix for the latent variables in

the state space model

#### Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

```
Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()
```

# Examples

```
beta <- matrix(
  data = c(
    0.7, 0.5, -0.1,
    0.0, 0.6, 0.4,
    0.0, 0.0, 0.5
),
  nrow = 3
)
psi <- 0.1 * diag(3)
lambda <- diag(3)
theta <- diag(3)
cov_eta <- SSMCovEta(
  beta = beta,
  psi = psi
)</pre>
```

SSMMeanEta 77

```
SSMCovY(
  lambda = lambda,
  theta = theta,
  cov_eta = cov_eta
)
```

SSMMeanEta

Steady-State Mean Vector for the Latent Variables in the State Space Model

# **Description**

The steady-state mean vector for the latent variables in the state space model Mean  $(\eta)$  is given by

$$Mean(\boldsymbol{\eta}) = (\mathbf{I} - \boldsymbol{\beta})^{-1} \boldsymbol{\alpha}$$

where  $\beta$  is the transition matrix relating the values of the latent variables at the previous to the current time point,  $\alpha$  is a vector of constant values for the dynamic model, and  $\mathbf{I}$  is an identity matrix.

#### Usage

```
SSMMeanEta(beta, alpha)
```

## **Arguments**

beta Numeric matrix. Transition matrix relating the values of the latent variables at

the previous to the current time point  $(\beta)$ .

alpha Numeric vector. Vector of constant values for the dynamic model  $(\alpha)$ .

## Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()

78 SSMMeanY

#### **Examples**

```
beta <- matrix(
  data = c(
    0.7, 0.5, -0.1,
    0.0, 0.6, 0.4,
    0.0, 0.0, 0.5
  ),
  nrow = 3
)
alpha <- rep(x = 1, times = 3)
SSMMeanEta(
  beta = beta,
  alpha = alpha
)</pre>
```

SSMMeanY

Steady-State Mean Vector for the Observed Variables in the State Space Model

# Description

The steady-state mean vector for the observed variables in the state space model  $\operatorname{Mean}(\eta)$  is given by

$$\nu + \Lambda \text{Mean}(\eta)$$

where  $\nu$  is the vector of intercept values for the measurement model,  $\Lambda$  is the matrix of factor loadings, and Mean  $(\eta)$  is the steady-state mean vector for the latent variables.

#### Usage

```
SSMMeanY(nu, lambda, mean_eta)
```

## **Arguments**

nu Numeric vector. Vector of intercept values for the measurement model  $(\nu)$ .

lambda Numeric matrix. Factor loading matrix linking the latent variables to the ob-

served variables  $(\Lambda)$ .

mean\_eta Numeric vector. Steady-state mean vector of the latent variables Mean  $(\eta)$ .

#### Author(s)

Ivan Jacob Agaloos Pesigan

TestPhi 79

#### See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability(), TestStationarity()

## **Examples**

```
beta <- matrix(</pre>
  data = c(
    0.7, 0.5, -0.1,
    0.0, 0.6, 0.4,
    0.0, 0.0, 0.5
  ),
  nrow = 3
)
alpha <- rep(x = 1, times = 3)
lambda <- diag(3)</pre>
nu \leftarrow rep(x = 1, times = 3)
mean_eta <- SSMMeanEta(</pre>
  beta = beta,
  alpha = alpha
)
SSMMeanY(
  nu = nu,
  lambda = lambda,
  mean_eta = mean_eta
)
```

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  $\Phi$  are less than zero.
- Test that the diagonal values of  $\Phi$  are between 0 to negative inifinity.

#### Usage

```
TestPhi(phi)
```

## **Arguments**

phi

Numeric matrix. The drift matrix  $(\Phi)$ .

80 TestStability

#### Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), 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)</pre>
```

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**

Х

Numeric matrix.

## Author(s)

Ivan Jacob Agaloos Pesigan

TestStationarity 81

#### See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), 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)</pre>
```

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**

Х

Numeric matrix.

#### Author(s)

Ivan Jacob Agaloos Pesigan

#### See Also

Other Simulation of State Space Models Data Functions: LinSDE2SSM(), LinSDECovEta(), LinSDECovY(), LinSDEMeanEta(), LinSDEMeanY(), ProjectToStability(), SSMCovEta(), SSMCovY(), SSMMeanEta(), SSMMeanY(), SimAlphaN(), SimBetaN2(), SimCovDiagN(), SimCovN(), SimIotaN(), SimPhiN(), SimSSMFixed(), SimSSMIVary(), SimSSMLinGrowth(), SimSSMLinGrowthIVary(), SimSSMLinSDEFixed(), SimSSMLinSDEIVary(), SimSSMOUFixed(), SimSSMOUIVary(), SimSSMVARFixed(), SimSSMVARIVary(), SpectralRadius(), TestPhi(), TestStability()

82 TestStationarity

# 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)</pre>
```

# **Index**

* Simulation of State Space Models Data	LinSDECovY, 11
Functions	LinSDEMeanEta, 12
LinSDE2SSM, 7	LinSDEMeanY, 13
LinSDECovEta, 9	SimPhiN, 27
LinSDECovY, 11	SimSSMLinSDEFixed, 45
LinSDEMeanEta, 12	SimSSMLinSDEIVary, 51
LinSDEMeanY, 13	TestPhi, 79
ProjectToStability, 19	TestStability, 80
SimAlphaN, 21	* methods
SimBetaN, 22	as.data.frame.simstatespace, 2
SimBetaN2, 23	as.matrix.simstatespace, 5
SimCovDiagN, 24	plot.simstatespace, 14
SimCovN, 25	print.simstatespace, 17
SimIotaN, 26	* <b>ou</b>
SimPhin, 27	SimSSMOUFixed, 55
SimSSMFixed, 28	SimSSMOUIVary, 61
SimSSMIVary, 33	* simStateSpace
SimSSMLinGrowth, 37	LinSDE2SSM, 7
SimSSMLinGrowthIVary, 41	LinSDECovEta, 9
SimSSMLinSDEFixed, 45	LinSDECovY, 11
SimSSMLinSDEIVary, 51	LinSDEMeanEta, 12
SimSSMOUFixed, 55	LinSDEMeanY, 13
SimSSMOUIVary, 61	ProjectToStability, 19
SimSSMVARFixed, 66	SimAlphaN, 21
SimSSMVARIVary, 69	SimBetaN, 22
SpectralRadius, 73	SimBetaN2, 23
SSMCovEta, 74	SimCovDiagN, 24
SSMCovY, 75	SimCovN, 25
SSMMeanEta, 76	SimIotaN, 26
SSMMeanY, 77	SimPhiN, 27
TestPhi, 79	SimSSMFixed, 28
TestStability, 80	SimSSMIVary, 33
TestStationarity, 81	SimSSMLinGrowth, 37
* growth	SimSSMLinGrowthIVary, 41
SimSSMLinGrowth, 37	SimSSMLinSDEFixed, 45
SimSSMLinGrowthIVary, 41	SimSSMLinSDEIVary, 51
* linsde	SimSSMOUFixed, 55
LinSDE2SSM, 7	SimSSMOUIVary, 61
LinSDECovEta, 9	SimSSMVARFixed, 66

84 INDEX

SimSSMVARIVary, 69	as.data.frame.simstatespace, 2
SpectralRadius, 73	as.matrix.simstatespace, 5
SSMCovEta, 74	
SSMCovY, 75	LinSDE2SSM, 7, 10, 11, 13, 14, 20–23, 25–28,
SSMMeanEta, 76	31, 35, 40, 43, 48, 53, 59, 63, 68, 71,
SSMMeanY, 77	73, 75–81
TestPhi, 79	LinSDECovEta, 9, 9, 11, 13, 14, 20–23, 25–28,
TestStability, 80	31, 35, 40, 43, 48, 53, 59, 63, 68, 71,
TestStationarity, 81	73, 75–81
* sim	LinSDECovY, 9, 10, 11, 13, 14, 20–23, 25–28,
SimSSMFixed, 28	31, 35, 40, 43, 48, 53, 59, 63, 68, 71,
SimSSMIVary, 33	73, 75–81
SimSSMLinGrowth, 37	LinSDEMeanEta, 9–11, 12, 14, 20–23, 25–28,
SimSSMLinGrowthIVary, 41	31, 35, 40, 43, 48, 53, 59, 63, 68, 71,
SimSSMLinSDEFixed, 45	73, 75–81
SimSSMLinSDEIVary, 51	LinSDEMeanY, 9-11, 13, 13, 20-23, 25-28, 31,
SimSSMOUFixed, 55	35, 40, 43, 48, 53, 59, 63, 68, 71, 73,
SimSSMOUIVary, 61	75–81
SimSSMVARFixed, 66	
SimSSMVARIVary, 69	plot.default(), 15
* ssm	plot.simstatespace, 14
ProjectToStability, 19	print.simstatespace, 17
SimAlphaN, 21	ProjectToStability, 9-11, 13, 14, 19,
SimBetaN, 22	21–23, 25–28, 31, 35, 40, 43, 48, 53,
SimBetaN2, 23	59, 63, 68, 71, 73, 75–81
SimCovDiagN, 24	ProjectToStability(), 23
SimCovN, 25	
SimIotaN, 26	SimAlphaN, 9–11, 13, 14, 20, 21, 22, 23,
SimSSMFixed, 28	25–28, 31, 35, 40, 43, 48, 53, 59, 63,
SimSSMIVary, 33	68, 71, 73, 75–81
SpectralRadius, 73	SimBetaN, 9-11, 13, 14, 20, 21, 22, 23, 25-28,
SSMCovEta, 74	31, 35, 40, 43, 48, 53, 59, 63, 68, 71,
SSMCovY, 75	73, 75–81
SSMMeanEta, 76	SimBetaN2, 9-11, 13, 14, 20-22, 23, 25-28,
SSMMeanY, 77	31, 35, 40, 43, 48, 53, 59, 63, 68, 71,
TestStationarity, 81	73, 75–81
* stability	SimCovDiagN, 9–11, 13, 14, 20–23, 24, 26–28,
ProjectToStability, 19	31, 35, 40, 43, 48, 53, 59, 63, 68, 71,
SpectralRadius, 73	73, 75–81
* test	SimCovN, 9–11, 13, 14, 20–23, 25, 25, 27, 28,
TestPhi, 79	31, 35, 40, 43, 48, 53, 59, 63, 68, 71,
TestStability, $80$	73, 75–81
TestStationarity, 81	SimIotaN, 9–11, 13, 14, 20–23, 25, 26, 26, 28,
* transformation	31, 35, 40, 43, 48, 53, 59, 63, 68, 71,
LinSDE2SSM, 7	73, 75–81
* var	SimPhiN, 9–11, 13, 14, 20–23, 25, 26, 27, 27,
SimSSMVARFixed, 66	31, 35, 40, 43, 48, 53, 59, 63, 68, 71,
SimSSMVARIVary, 69	73, 75–81

INDEX 85

```
SimSSMFixed, 9–11, 13, 14, 20–23, 25–27, 28,
         28, 35, 40, 43, 48, 53, 59, 63, 68, 71,
         73, 75–81
SimSSMFixed(), 34
SimSSMIVary, 9-11, 13, 14, 20-23, 25-28, 31,
         33, 40, 43, 48, 53, 59, 63, 68, 71, 73,
         75-81
SimSSMLinGrowth, 9–11, 13, 14, 20–23,
         25-28, 31, 35, 37, 43, 48, 53, 59, 63,
         68, 71, 73, 75–81
SimSSMLinGrowth(), 42
SimSSMLinGrowthIVary, 9–11, 13, 14, 20–23,
         25–28, 31, 35, 40, 41, 48, 53, 59, 63,
         68, 71, 73, 75-81
SimSSMLinSDEFixed, 9–11, 13, 14, 20–23,
         25–28, 31, 35, 40, 43, 45, 53, 59, 63,
         68, 71, 73, 75–81
SimSSMLinSDEFixed(), 52
SimSSMLinSDEIVary, 9–11, 13, 14, 20–23,
         25-28, 31, 35, 40, 43, 48, 51, 59, 63,
         68, 71, 73, 75–81
SimSSMOUFixed, 9–11, 13, 14, 20–23, 25–28,
         31, 35, 40, 43, 48, 53, 55, 63, 68, 71,
         73, 75–81
SimSSMOUFixed(), 62
SimSSMOUIVary, 9-11, 13, 14, 20-23, 25-28,
         31, 35, 40, 43, 48, 53, 59, 61, 68, 71,
         73, 75–81
SimSSMVARFixed, 9–11, 13, 14, 20–23, 25–28,
         31, 35, 40, 43, 48, 53, 59, 63, 66, 71,
         73, 75–81
SimSSMVARFixed(), 70
SimSSMVARIVary, 9-11, 13, 14, 20-23, 25-28,
         31, 35, 40, 43, 48, 53, 59, 63, 68, 69,
         73, 75–81
SpectralRadius, 9–11, 13, 14, 20–23, 25–28,
         31, 35, 40, 43, 48, 53, 59, 63, 68, 71,
         73, 75–81
SSMCovEta, 9–11, 13, 14, 20–23, 25–28, 31,
         35, 40, 43, 48, 53, 59, 63, 68, 71, 73,
         74, 76–81
SSMCovY, 9–11, 13, 14, 20–23, 25–28, 31, 35,
         40, 43, 48, 53, 59, 63, 68, 71, 73, 75,
         75, 77–81
SSMMeanEta, 9–11, 13, 14, 20–23, 25–28, 31,
         35, 40, 43, 48, 53, 59, 63, 68, 71, 73,
         75, 76, 76, 78–81
SSMMeanY, 9-11, 13, 14, 20-23, 25-28, 31, 35,
```

```
40, 43, 48, 53, 59, 63, 68, 71, 73, 75, \\ 76, 77, 77, 79-81
\text{TestPhi, } 9-11, 13, 14, 20-23, 25-28, 31, 35, \\ 40, 43, 48, 53, 59, 63, 68, 71, 73, \\ 75-78, 79, 80, 81
\text{TestPhi(), } 27
\text{TestStability, } 9-11, 13, 14, 20-23, 25-28, \\ 31, 35, 40, 43, 48, 53, 59, 63, 68, 71, \\ 73, 75-79, 80, 81
\text{TestStationarity, } 9-11, 13, 14, 20-23, \\ 25-28, 31, 35, 40, 43, 48, 53, 59, 63, \\ 68, 71, 73, 75-80, 81
\text{TestStationarity(), } 22
```