Package 'varmapack'

November 19, 2026

Type Package

Title Burn-in-Free Simulation and Analysis of Gaussian VARMA Models

Version 0.2.0

Description Simulates Gaussian vector autoregressive-moving-average time-series models without a burn-in period by drawing startup shocks from their model-implied conditional distribution. Also simulates stationary Ornstein-Uhlenbeck and Slepian-based moving-average models at unevenly spaced observation times. Provides model test cases, autocovariances, spectral radii, impulse responses, and innovation covariance recovery.

License MIT + file LICENSE

Encoding UTF-8

Depends R (>= 4.0.0)

Imports R6, randompack (>= 0.1.10)

LinkingTo randompack

Suggests knitr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.2

URL https://github.com/jonasson2/varmapack

BugReports https://github.com/jonasson2/varmapack/issues

NeedsCompilation yes

Author Kristján Jónasson [aut, cre]

Maintainer Kristján Jónasson <jonasson@hi.is>

Repository CRAN

Date/Publication 2026-11-19 16:20:02 UTC

Contents

varmapack_autocov 2
varmapack_cov2corr 3
varmapack_cov2sig 3
varmapack_gamma_times 4
varmapack_model 5
varmapack_ou_model 6
varmapack_slepian_model 7
varmapack_testcase 8
varmapack_testcases 9

Index **10**

`varmapack_autocov` *Sample autocovariances of an observed series*

Description

Compute sample autocovariance matrices up to a specified lag for a numeric time-series matrix with variables in rows and observations in columns.

Usage

```
varmapack_autocov(X, maxlag, norm = "ML")
```

Arguments

`X`	Numeric `r` by `n` observed time-series matrix.
`maxlag`	Largest lag to compute, between zero and `n - 1`.
`norm`	Either `"ML"`, which divides every lag by `n`, or `"C"`, which divides the lag-`k` covariance by `n - k`.

Value

An `r` by `r` by `maxlag + 1` array. Its `k + 1` plane estimates `Cov(x_t, x_{t-k})`.

Examples

```
X <- rbind(1:10, (1:10)^2)
varmapack_autocov(X, maxlag = 2)
```

varmapack_cov2corr *Convert covariances to correlations*

Description

Convert a covariance matrix or sequence of autocovariance matrices to correlations using the marginal standard deviations at lag zero.

Usage

```
varmapack_cov2corr(cov)
```

Arguments

cov	A finite numeric r by r matrix or r by r by maxlag + 1 array. The diagonal entries of its lag-zero matrix must be positive.

Value

A numeric array with the same dimensions as cov. Lag-zero diagonal entries are exactly one. Other entries are not clipped to [-1,1].

Examples

```
cov <- array(c(4, 3, 3, 9, 2, 6, -3, 1.5), dim = c(2, 2, 2))
varmapack_cov2corr(cov)
```

varmapack_cov2sig *Recover innovation covariance from model covariance*

Description

Recover the innovation covariance with the AR and MA coefficients held fixed. The AR model must be stationary, and the target and recovered covariances must be positive semidefinite; singular covariances are allowed.

Usage

```
varmapack_cov2sig(cov, A = NULL, B = NULL)
```

Arguments

cov	Numeric r by r lag-zero model covariance matrix. Only its lower triangle is used.
A, B	AR and MA coefficient matrices, as r by r matrices or r by r by order arrays. NULL omits the corresponding terms.

Value

A symmetric `r` by `r` innovation covariance matrix. Inputs are unchanged. An incompatible target or unreliable inverse raises an error.

See Also

`varmapack_model()`, `varmapack_cov2corr()`

Examples

```
Sig <- varmapack_cov2sig(matrix(4/3, 1, 1), A = matrix(0.5, 1, 1))
varmapack_model(A = matrix(0.5, 1, 1), Sig = Sig)$acvf(0)
```

`varmapack_gamma_times` *Generate observation times with Gamma spacing*

Description

Generate observation times with Gamma spacing

Usage

```
varmapack_gamma_times(n, shape = 16, mean_gap = 1, rng = NULL)
```

Arguments

`n`	Number of times, at least one.
`shape`	Positive Gamma shape. `Inf` gives constant spacing.
`mean_gap`	Positive expected gap, in the same units as OU persistence.
`rng`	Optional `randompack::randompack_rng()` object.

Value

A strictly increasing numeric vector starting at zero. The `n-1` independent gaps have scale `mean_gap/shape` and coefficient of variation `1/sqrt(shape)`. Their realized mean is not normalized. Unrepresentable positive gaps or times produce an error.

Examples

```
t <- varmapack_gamma_times(100, shape = 4, mean_gap = 1)
```

varmapack_model *Create a VARMA or VARMAX model*

Description

Create a model object for simulation of a Gaussian VAR, VMA, VARMA, or VARMAX time series. The object stores model parameters and provides a `$sim()` method for generating one or more independent series.

Usage

```
varmapack_model(A = NULL, B = NULL, C = NULL, Sig, mu = NULL)
```

Arguments

`A`	Autoregressive coefficient matrices as an `r` by `r` matrix or an `r` by `r` by `p` array. `NULL` gives no autoregressive terms.
`B`	Moving-average coefficient matrices as an `r` by `r` matrix or an `r` by `r` by `q` array. `NULL` gives no moving-average terms.
`C`	Exogenous coefficient matrices as an `r` by `d` matrix or an `r` by `d` by `s` array. `NULL` gives a VARMA model rather than VARMAX.
`Sig`	`r` by `r` innovation covariance matrix.
`mu`	Optional `r` vector or `r` by `nmu` matrix of time-series means. The last mean vector repeats for the rest of each series. Means are not supported for VARMAX models.

Value

A `VarmapackModel` object.

Simulation

`model$sim(length, nrep = 1L, X0 = NULL, z = NULL, rng = NULL, return_shocks = FALSE)` simulates `nrep` independent series of the supplied `length`.

For VARMA models, `X0` is optional. If provided, it is an `r` by `nX0` matrix or an `r` by `nX0` by `nrep` array. Its second dimension must be at least `max(p, q)`, and its third dimension, when present, must be 1 or `nrep`. Nonstationary VARMA models require `X0`; with MA terms, `Sig` must be positive definite and startup shocks are conditioned on the residual equations implied by the supplied history.

For VARMAX models, `z` is required. `X0` has the same form and its second dimension must be at least `max(p, q, s - 1)`. It may be omitted when this minimum is zero. The exogenous input `z` is a `d` by `length` matrix or a `d` by `length` by `nrep` array. Its third dimension, when present, must be 1 or `nrep`.

Pass a `randompack::randompack_rng()` object through `rng` to control the random stream. When it is omitted, Varmapack uses a temporary randomized default Randompack generator. With `return_shocks = FALSE`, the result is an `r` by `length` by `nrep` array. With `return_shocks = TRUE`, it is a list with components `X` and `E`, each with that shape.

Model analysis

`model$acvf(maxlag)` returns theoretical VARMA autocovariances, `model$psi(maxlag)` returns impulse-response matrices, and `model$irf(maxlag)` returns orthogonalized impulse-response matrices. `model$specrad()` and `model$ma_specrad()` return the AR and MA spectral radii, respectively. Theoretical autocovariances are not available for VARMAX models with exogenous terms.

Examples

```
A <- matrix(c(0.4, 0.1, -0.2, 0.3), 2, 2)
model <- varmapack_model(A = A, Sig = diag(2))
X <- model$sim(100)

rng <- randompack::randompack_rng()
rng$seed(123)
X <- model$sim(100, nrep = 3, rng = rng)
```

---

varmapack_ou_model *Create a stationary Ornstein–Uhlenbeck model*

Description

A Gaussian continuous-time model, simulated exactly at arbitrary observation times without burn-in. With diagonal drift, component i has autocorrelation `exp(-gap/tau[i])`.

Usage

```
varmapack_ou_model(tau = NULL, cov, mu = NULL, drift = NULL)
```

Arguments

`tau`	Finite positive numeric vector of persistence times. Supply exactly one of `tau` and `drift`.
`cov`	Stationary process covariance, not innovation covariance. Only the lower triangle is used. Both `cov` and the driving covariance `K %*% cov + cov %*% t(K)` must be positive semidefinite, where `K = diag(1/tau)` for persistence-time initialization. Singular covariances are allowed.
`mu`	Optional constant mean vector, defaulting to zero.
`drift`	Square matrix K in the drift term `-K %*% (X - mu)`. Its eigenvalues must have strictly positive real parts.

Details

Over a gap of length d, the autoregressive coefficient matrix is the matrix exponential $\exp(-Kd)$. With constant gaps, the sampled process is a VAR(1); with unequal gaps, the matrix varies from step to step.

Value

A VarmapackOUModel object with fields tau, drift, cov, mu, r, and method sim(t, nrep = 1L, rng = NULL). The result has dimensions r by length(t) by nrep, including for one-dimensional models. The unused parameter field (tau or drift) is NULL. Times must be finite and strictly increasing. The first observation is drawn from N(mu, cov). Supply a randompack::randompack_rng() through rng to control the stream. acvf(lags) returns Cov(X(t+h), X(t)) as an r by r by length(lags) array. specrad(gap) returns the sampled VAR(1) spectral radius at a fixed gap.

Examples

```
rng <- randompack::randompack_rng()
rng$seed(123)
t <- varmapack_gamma_times(100, shape = 16, rng = rng)
model <- varmapack_ou_model(c(1, 2), matrix(c(1, .5, .5, 1), 2))
X <- model$sim(t, nrep = 3, rng = rng)
```

---

varmapack_slepian_model

Create a stationary Slepian-based moving-average model

Description

A stationary Gaussian model with unit-width Brownian noise windows. It reduces to a vector MA(1) at unit spacing and can be simulated exactly at arbitrary observation times without burn-in.

Usage

```
varmapack_slepian_model(theta, cov, mu = NULL)
```

Arguments

theta	Real square unit-step MA(1) coefficient matrix.
cov	Stationary process covariance; only the lower triangle is used. The driving covariance Sig must be positive semidefinite and determined by cov = Sig + theta %*% Sig %*% t(theta).
mu	Optional constant mean vector, defaulting to zero.

Value

A VarmapackSlepianModel with fields theta, cov, mu, r, and method sim(t, nrep = 1L, rng = NULL). The result has dimensions r by length(t) by nrep. Times must be finite and strictly increasing. Supply a randompack::randompack_rng() through rng to control the stream. acvf(lags) returns Cov(X(t+h), X(t)) as an r by r by length(lags) array.

Examples

```
rng <- randompack::randompack_rng()
rng$seed(123)
t <- varmapack_gamma_times(100, shape = 16, rng = rng)
model <- varmapack_slepian_model(matrix(.5), matrix(1.25))
X <- model$sim(t, nrep = 3, rng = rng)
```

`varmapack_testcase` *Create a VARMA testcase model*

Description

Create a built-in named testcase or an unnamed random, deterministic, or spectral-radius-controlled VARMA testcase.

Usage

```
varmapack_testcase(
  which = "random",
  p = NULL,
  q = NULL,
  r = NULL,
  rho = 0,
  rng = NULL
)
```

Arguments

`which`	A built-in testcase name, a one-based built-in testcase index, or one of `"random"`, `"deterministic"`, and `"rho"`.
`p`, `q`, `r`	Required AR order, MA order, and series dimension for unnamed testcases.
`rho`	Target spectral radius when `which = "rho"`.
`rng`	Optional `randompack::randompack_rng()` used for a random testcase.

Value

A `VarmapackModel` object.

Examples

```
model <- varmapack_testcase("smallARMA1")
model <- varmapack_testcase("rho", p = 3, q = 1, r = 2, rho = 0.8)
```

varmapack_testcases *List named VARMA testcases*

Description

List named VARMA testcases

Usage

```
varmapack_testcases()
```

Value

A data frame with columns `index`, `name`, `p`, `q`, and `r`.

Examples

```
varmapack_testcases()
```

Index

varmapack_autocov, 2
varmapack_cov2corr, 3
varmapack_cov2corr(), *4*
varmapack_cov2sig, 3
varmapack_gamma_times, 4
varmapack_model, 5
varmapack_model(), *4*
varmapack_ou_model, 6
varmapack_slepian_model, 7
varmapack_testcase, 8
varmapack_testcases, 9