Package ‘simAutoReg’

July 26, 2023

Title Simulate Data from Autoregressive Models

Version 0.9.1

Description Simulate data from autoregressive models.

URL `https://github.com/ijapesigan/simAutoReg`, `https://ijapesigan.github.io/simAutoReg/`

BugReports `https://github.com/ijapesigan/simAutoReg/issues`

License GPL (>= 3)

Encoding UTF-8

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Depends R (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp

Suggests knitr, rmarkdown, testthat, vars

RoxygenNote 7.2.3

NeedsCompilation yes

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

Maintainer Ivan Jacob Agaloos Pesigan <`ijapesigan@gmail.com`>

R topics documented:

FitVAROLS *Fit Vector Autoregressive (VAR) Model Parameters using OLS*

Description

This function estimates the parameters of a VAR model using the Ordinary Least Squares (OLS) method. The OLS method is used to estimate the autoregressive and cross-regression coefficients.

Usage

```
FitVAROLS(Y, X)
```

Arguments

`Y`	Numeric matrix. Matrix of dependent variables (Y).
`X`	Numeric matrix. Matrix of lagged predictors (X).

Details

The `FitVAROLS` function estimates the parameters of a Vector Autoregressive (VAR) model using the Ordinary Least Squares (OLS) method. Given the input matrices `Y` and `X`, where `Y` is the matrix of dependent variables, and `X` is the matrix of lagged predictors, the function computes the autoregressive and cross-regression coefficients of the VAR model.

The steps involved in estimating the VAR model parameters using OLS are as follows:

- Compute the QR decomposition of the lagged predictor matrix `X` using the `qr_econ` function from the Armadillo library.
- Extract the `Q` and `R` matrices from the QR decomposition.
- Solve the linear system `R * coef = Q.t() * Y` to estimate the VAR model coefficients `coef`.

The function returns a matrix containing the estimated autoregressive and cross-regression coefficients of the VAR model.

Value

Matrix of estimated autoregressive and cross-regression coefficients.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

The `qr_econ` function from the Armadillo library for QR decomposition.

Examples

```
set.seed(42)
time <- 100000L
burn_in <- 200
k <- 3
p <- 2
constant <- c(1, 1, 1)
coef <- matrix(
  data = c(
    0.4, 0.0, 0.0, 0.1, 0.0, 0.0,
    0.0, 0.5, 0.0, 0.0, 0.2, 0.0,
    0.0, 0.0, 0.6, 0.0, 0.0, 0.3
  ),
  nrow = k,
  byrow = TRUE
)
chol_cov <- chol(
  matrix(
    data = c(
      0.1, 0.0, 0.0,
      0.0, 0.1, 0.0,
      0.0, 0.0, 0.1
    ),
    nrow = k,
    byrow = TRUE
  )
)
y <- SimVAR(
  time = time,
  burn_in = burn_in,
  constant = constant,
  coef = coef,
  chol_cov = chol_cov
)
yx <- YX(y, p)
FitVAROLS(Y = yx$Y, X = yx$X)
```

SimAR — *Simulate Data from an Autoregressive Model with Constant Term*

Description

This function generates synthetic time series data from an autoregressive (AR) model.

Usage

```
SimAR(time, burn_in, constant, coef, sd)
```

Arguments

`time`	Integer. Number of time points to simulate.
`burn_in`	Integer. Number of burn-in periods before recording data.
`constant`	Numeric. The constant term of the AR model.
`coef`	Numeric vector. Autoregressive coefficients.
`sd`	Numeric. The standard deviation of the random noise.

Details

The `SimAR` function generates synthetic time series data from an autoregressive (AR) model. The generated data follows the AR(p) model, where `p` is the number of coefficients specified in `coef`. The generated time series data includes a constant term and autoregressive terms based on the provided coefficients. Random noise, sampled from a normal distribution with mean 0 and standard deviation `sd`, is added to the time series. Additionally, a burn-in period can be specified to exclude initial data points from the output.

The steps in generating the autoregressive time series with burn-in are as follows:

- Set the order of the AR model to `p`.
- Generate random noise from a normal distribution with mean 0 and standard deviation `sd`.
- Generate the autoregressive time series with burn-in using the formula:

  $$Y_t = constant + \sum_{i=1}^{p} (coef[i] * Y_{t-i}) + noise_t$$

  where Y_t is the time series data at time t, $constant$ is the constant term, $coef[i]$ are the autoregressive coefficients, Y_{t-i} are the lagged data points up to order `p`, and $noise_t$ is the random noise at time t.
- Optionally, remove the burn-in period from the generated time series data.

Value

Numeric vector containing the simulated time series data.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
set.seed(42)
SimAR(time = 10, burn_in = 5, constant = 2, coef = c(0.5, -0.3), sd = 0.1)
```

SimMVN *Simulate Multivariate Normal Random Numbers*

Description

This function generates multivariate normal random numbers.

Usage

```
SimMVN(n, location, chol_scale)
```

Arguments

`n`	Integer. Number of samples to generate.
`location`	Numeric vector. Mean vector of length k, where k is the number of variables.
`chol_scale`	Numeric matrix. Cholesky decomposition of the covariance matrix of dimensions k x k.

Details

The `SimMVN` function generates multivariate normal random numbers using the Cholesky decomposition method. Given the number of samples `n`, the mean vector `location` of length `k` (where `k` is the number of variables), and the Cholesky decomposition `chol_scale` of the covariance matrix of dimensions k x k, the function produces a matrix of multivariate normal random numbers.

The steps involved in generating multivariate normal random numbers are as follows:

- Determine the number of variables `k` from the length of the mean vector.
- Generate random data from a standard multivariate normal distribution, resulting in an n x k matrix of random numbers.
- Transform the standard normal random data into multivariate normal random data using the Cholesky decomposition `chol_scale`.
- Add the mean vector `location` to the transformed data to obtain the final simulated multivariate normal random numbers.

The function returns a matrix of simulated multivariate normal random numbers with dimensions n x k, where n is the number of samples and k is the number of variables. This matrix can be used for various statistical analyses and simulations.

Value

Matrix containing the simulated multivariate normal random numbers, with dimensions n x k, where n is the number of samples and k is the number of variables.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

The chol function in R to obtain the Cholesky decomposition of a covariance matrix.

Examples

```
set.seed(42)
n <- 100000L
location <- c(0.5, -0.2, 0.1)
scale <- matrix(
  data = c(1.0, 0.3, 0.3, 0.3, 1.0, 0.2, 0.3, 0.2, 1.0),
  nrow = 3,
  byrow = TRUE
)
chol_scale <- chol(scale)
y <- SimMVN(n = n, location = location, chol_scale = chol_scale)
colMeans(y)
var(y)
```

SimVAR *Simulate Data from a Vector Autoregressive (VAR) Model*

Description

This function generates synthetic time series data from a Vector Autoregressive (VAR) model.

Usage

```
SimVAR(time, burn_in, constant, coef, chol_cov)
```

Arguments

time	Integer. Number of time points to simulate.
burn_in	Integer. Number of burn-in observations to exclude before returning the results.
constant	Numeric vector. The constant term vector of length k, where k is the number of variables.
coef	Numeric matrix. Coefficient matrix with dimensions k x (k * p). Each k x k block corresponds to the coefficient matrix for a particular lag.
chol_cov	Numeric matrix. The Cholesky decomposition of the covariance matrix of the multivariate normal noise. It should have dimensions k x k.

Details

The `SimVAR` function generates synthetic time series data from a Vector Autoregressive (VAR) model. The VAR model is defined by the constant term `constant`, the coefficient matrix `coef`, and the Cholesky decomposition of the covariance matrix of the multivariate normal noise `chol_cov`. The generated time series data follows a VAR(p) process, where `p` is the number of lags specified by the size of `coef`. The generated data includes a burn-in period, which is excluded before returning the results.

The steps involved in generating the VAR time series data are as follows:

- Extract the number of variables `k` and the number of lags `p` from the input.
- Create a matrix `data` of size `k x (time + burn_in)` to store the generated VAR time series data.
- Set the initial values of the matrix `data` using the constant term `constant`.
- For each time point starting from the `p`-th time point to `time + burn_in - 1`:
- Generate a vector of random noise from a multivariate normal distribution with mean 0 and covariance matrix `chol_cov`.
- Generate the VAR time series values for each variable `j` at time `i` using the formula:

$$Y_{ij} = constant_j + \sum_{l=1}^{p} \sum_{m=1}^{k} (coef_{jm} * Y_{im}) + \text{noise}_j$$

  where Y_{ij} is the value of variable `j` at time `i`, `constant_j` is the constant term for variable `j`, `coef_{jm}` are the coefficients for variable `j` from lagged variables up to order `p`, Y_{im} are the lagged values of variable `m` up to order `p` at time `i`, and `noise_{j}` is the element `j` from the generated vector of random noise.
- Transpose the matrix `data` and return only the required time period after the burn-in period, which is from column `burn_in` to column `time + burn_in - 1`.

Value

Numeric matrix containing the simulated time series data with dimensions k x (time - burn_in), where `k` is the number of variables and time is the number of observations.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
set.seed(42)
time <- 100000L
burn_in <- 200
k <- 3
p <- 2
constant <- c(1, 1, 1)
coef <- matrix(
  data = c(
```

```
    0.4, 0.0, 0.0, 0.1, 0.0, 0.0,
    0.0, 0.5, 0.0, 0.0, 0.2, 0.0,
    0.0, 0.0, 0.6, 0.0, 0.0, 0.3
  ),
  nrow = k,
  byrow = TRUE
)
chol_cov <- chol(
  matrix(
    data = c(
      0.1, 0.0, 0.0,
      0.0, 0.1, 0.0,
      0.0, 0.0, 0.1
    ),
    nrow = k,
    byrow = TRUE
  )
)
y <- SimVAR(
  time = time,
  burn_in = burn_in,
  constant = constant,
  coef = coef,
  chol_cov = chol_cov
)
head(y)
```

`SimVariance` *Generate Random Data for the Variance Vector*

Description

This function generates random data for the variance vector given by

$$\boldsymbol{\sigma^2} = \exp\left(\boldsymbol{\mu} + \boldsymbol{\varepsilon}\right) \quad \text{with} \boldsymbol{\varepsilon} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{\Sigma}\right)$$

.

Usage

```
SimVariance(n, location, chol_scale)
```

Arguments

`n`	Integer. Number of samples to generate.
`location`	Numeric vector. The constant term $\boldsymbol{\mu}$.
`chol_scale`	Numeric matrix. Cholesky decomposition of the covariance matrix $\boldsymbol{\Sigma}$ for the multivariate normal random error $\boldsymbol{\varepsilon}$.

Value

Matrix with each row containing the simulated variance vector for each sample.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
set.seed(42)
n <- 100
location <- c(0.5, -0.2, 0.1)
chol_scale <- chol(
  matrix(
    data = c(1.0, 0.3, 0.3, 0.3, 1.0, 0.2, 0.3, 0.2, 1.0),
    nrow = 3,
    byrow = TRUE
  )
)
SimVariance(n = n, location = location, chol_scale = chol_scale)
```

YX *Create Y and X Matrices*

Description

This function creates the Y and X matrices.

Usage

```
YX(data, p)
```

Arguments

`data`	Numeric matrix. The time series data with dimensions n x k, where n is the number of observations and k is the number of variables.
`p`	Integer. The order of the VAR model (number of lags).

Details

The `YX` function creates the Y and X matrices required for fitting a Vector Autoregressive (VAR) model. Given the input `data` matrix with dimensions n x k, where n is the number of observations and k is the number of variables, and the order of the VAR model `p` (number of lags), the function constructs lagged predictor matrix X and the dependent variable matrix Y. The matrices X and Y are used as inputs for estimating the VAR model parameters.

The steps involved in creating the Y and X matrices are as follows:

- Determine the number of observations n and the number of variables k from the input data matrix.
- Create matrices X and Y to store lagged variables and the dependent variable, respectively.
- Populate the matrices X and Y with the appropriate lagged data. The predictors matrix X contains the lagged values of the dependent variables, while the dependent variable matrix Y contains the original values of the dependent variables.

The function returns a list containing the Y and X matrices, which can be used for further analysis and estimation of the VAR model parameters.

Value

List containing the Y and X matrices.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

The `SimVAR` function for simulating time series data from a VAR model.

Examples

```
set.seed(42)
time <- 100000L
burn_in <- 200
k <- 3
p <- 2
constant <- c(1, 1, 1)
coef <- matrix(
  data = c(
    0.4, 0.0, 0.0, 0.1, 0.0, 0.0,
    0.0, 0.5, 0.0, 0.0, 0.2, 0.0,
    0.0, 0.0, 0.6, 0.0, 0.0, 0.3
  ),
  nrow = k,
  byrow = TRUE
)
chol_cov <- chol(
  matrix(
    data = c(
      0.1, 0.0, 0.0,
      0.0, 0.1, 0.0,
      0.0, 0.0, 0.1
    ),
    nrow = k,
    byrow = TRUE
  )
)
y <- SimVAR(
```

```
    time = time,
    burn_in = burn_in,
    constant = constant,
    coef = coef,
    chol_cov = chol_cov
  )
  yx <- YX(data = y, p = p)
  str(yx)
```

Index