Package 'simAutoReg'

July 31, 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

LazyData true

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Depends R (>= 3.5.0)

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp

Suggests knitr, rmarkdown, testthat

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:

`FitVARLasso` *Fit Vector Autoregressive (VAR) Model Parameters using Lasso Regularization*

Description

This function estimates the parameters of a VAR model using the Lasso regularization method with cyclical coordinate descent. The Lasso method is used to estimate the autoregressive and cross-regression coefficients with sparsity.

Usage

```
FitVARLasso(Y_std, X_std, lambda, max_iter = 10000L, tol = 1e-05)
```

Arguments

`Y_std`	Numeric matrix. Matrix of standardized dependent variables (Y).
`X_std`	Numeric matrix. Matrix of standardized predictors (X).
`lambda`	Lasso hyperparameter. The regularization strength controlling the sparsity.
`max_iter`	Integer. The maximum number of iterations for the coordinate descent algorithm. Default is 10000.
`tol`	Numeric. Convergence tolerance. The algorithm stops when the change in coefficients between iterations is below this tolerance. Default is 1e-6.

Details

The `FitVARLasso()` function estimates the parameters of a Vector Autoregressive (VAR) model using the Lasso regularization method. Given the input matrices `Y_std` and `X_std`, where `Y_std` is the matrix of standardized dependent variables, and `X_std` is the matrix of standardized predictors, the function computes the autoregressive and cross-regression coefficients of the VAR model with sparsity induced by the Lasso regularization.

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

- **Initialization**: The function initializes the coefficient matrix `beta` with OLS estimates. The `beta` matrix will store the estimated autoregressive and cross-regression coefficients.

- **Coordinate Descent Loop**: The function performs the cyclical coordinate descent algorithm to estimate the coefficients iteratively. The loop iterates `max_iter` times (default is 10000), or until convergence is achieved. The outer loop iterates over the predictor variables (columns of `X_std`), while the inner loop iterates over the outcome variables (columns of `Y_std`).
- **Coefficient Update**: For each predictor variable (column of `X_std`), the function iteratively updates the corresponding column of `beta` using the coordinate descent algorithm with L1 norm regularization (Lasso). The update involves calculating the soft-thresholded value `c`, which encourages sparsity in the coefficients. The algorithm continues until the change in coefficients between iterations is below the specified tolerance `tol` or when the maximum number of iterations is reached.
- **Convergence Check**: The function checks for convergence by comparing the current `beta` matrix with the previous iteration's `beta_old`. If the maximum absolute difference between `beta` and `beta_old` is below the tolerance `tol`, the algorithm is considered converged, and the loop exits.

Value

Matrix of estimated autoregressive and cross-regression coefficients.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

The `FitVAROLS()` function for estimating VAR model parameters using OLS.

Examples

```
Y_std <- StdMat(VAR_YX$Y)
X_std <- StdMat(VAR_YX$X[, -1])
lambda <- 58.57
FitVARLasso(Y_std = Y_std, X_std = X_std, lambda = lambda)
```

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 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 predictors, the function computes the autoregressive and cross-regression coefficients of the VAR model. Note that if the first column of X is a vector of ones, the constant vector is also estimated.

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 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 function from the Armadillo library for QR decomposition.

Examples

```
FitVAROLS(Y = VAR_YX$Y, X = VAR_YX$X)
```

OrigScale *Return Standardized Estimates to the Original Scale*

Description

Return Standardized Estimates to the Original Scale

Usage

```
OrigScale(coef_std, sd_Y, sd_X, mean_Y, mean_X)
```

Arguments

coef_std	Numeric matrix. Standardized estimates of the autoregression and cross regression coefficients.
sd_Y	Numeric vector. Standard deviations of the Y matrix.
sd_X	Numeric vector. Standard deviations of the X matrix.
mean_Y	Numeric vector. Means of the Y matrix.
mean_X	Numeric vector. Means of the X matrix.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
coef_std <- FitVAROLS(Y = StdMat(VAR_YX$Y), X = StdMat(VAR_YX$X[, -1]))
sd_Y <- sqrt(diag(var(VAR_YX$Y)))
sd_X <- sqrt(diag(var(VAR_YX$X[, -1])))
mean_Y <- colMeans(VAR_YX$Y)
mean_X <- colMeans(VAR_YX$X[, -1])
OrigScale(coef_std = coef_std, sd_Y = sd_Y, sd_X = sd_X, mean_Y = mean_Y, mean_X = mean_X)
FitVAROLS(Y = VAR_YX$Y, X = VAR_YX$X[, -1])
```

SearchVARLasso *Compute AIC, BIC, and EBIC for Lasso Regularization*

Description

This function computes the Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), and Extended Bayesian Information Criterion (EBIC) for a given matrix of predictors X, a matrix of outcomes Y, and a vector of lambda hyperparameters for Lasso regularization.

Usage

```
SearchVARLasso(Y_std, X_std, lambdas, max_iter = 10000L, tol = 1e-05)
```

Arguments

Y_std	Numeric matrix. Matrix of standardized dependent variables (Y).
X_std	Numeric matrix. Matrix of standardized predictors (X).
lambdas	Numeric vector. Vector of lambda hyperparameters for Lasso regularization.
max_iter	Integer. The maximum number of iterations for the coordinate descent algorithm. Default is 10000.
tol	Numeric. Convergence tolerance. The algorithm stops when the change in coefficients between iterations is below this tolerance. Default is 1e-5.

Value

List containing two elements:

- Element 1: Matrix with columns for lambda, AIC, BIC, and EBIC values.
- Element 2: List of matrices containing the estimated autoregressive and cross-regression coefficients for each lambda.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
Y_std <- StdMat(VAR_YX$Y)
X_std <- StdMat(VAR_YX$X[, -1])
lambdas <- 10^seq(-5, 5, length.out = 100)
search <- SearchVARLasso(Y_std = Y_std, X_std = X_std, lambdas = lambdas)
plot(x = 1:nrow(search$criteria), y = search$criteria[, 4], type = "b", xlab = "lambda", ylab = "EBIC")
```

SelectVARLasso *Select the Lasso Estimates from the Grid Search*

Description

Select the Lasso Estimates from the Grid Search

Usage

```
SelectVARLasso(search, crit = "ebic")
```

Arguments

search	Object. Output of the SearchVARLasso() function.
crit	Character string. Information criteria to use. Valid values include "aic", "bic", and "ebic".

Value

Returns the Lasso estimates of autoregression and cross regression coefficients.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
Y_std <- StdMat(VAR_YX$Y)
X_std <- StdMat(VAR_YX$X[, -1])
lambdas <- 10^seq(-5, 5, length.out = 100)
search <- SearchVARLasso(Y_std = Y_std, X_std = X_std, lambdas = lambdas)
SelectVARLasso(search, crit = "ebic")
```

`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 process 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. A burn-in period is 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` based on the length of `coef`.
- Create a vector data of size `time` + `burn_in` to store the generated AR time series data.
- Create a vector data of size `time` + `burn_in` of random process 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.

- 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 by 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` by `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` by `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` by `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` by `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 <- 1000L
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 by (k * p). Each k by 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 by 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 process 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 by (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 t using the formula:

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

where Y_{tj} is the value of variable `j` at time `t`, $constant_j$ is the constant term for variable `j`, $coef_{jm}$ are the coefficients for variable `j` from lagged variables up to order `p`, Y_{tm} are the lagged values of variable `m` up to order `p` at time `t`, and $noise_j$ is the element `j` from the generated vector of random process 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` by `(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 <- 50L
burn_in <- 10L
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(diag(3))
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

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

.

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 $\boldsymbol{\varepsilon}$.

Details

The `SimVariance()` function generates random data for the variance vector based on the exponential of a multivariate normal distribution. Given the number of samples `n`, the constant term $\boldsymbol{\mu}$ represented by the `location` vector, and the Cholesky decomposition matrix $\boldsymbol{\Sigma}$ for the multivariate normal random error $\boldsymbol{\varepsilon}$, the function simulates n independent samples of the variance vector $\boldsymbol{\sigma^2}$. Each sample of the variance vector $\boldsymbol{\sigma^2}$ is obtained by calculating the exponential of random variations to the mean vector $\boldsymbol{\mu}$. The random variations are generated using the Cholesky decomposition of the covariance matrix $\boldsymbol{\Sigma}$. Finally, the function returns a matrix with each column containing the simulated variance vector for each sample.

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)
```

SimVARZIP *Simulate Data from a Vector Autoregressive Zero-Inflated Poisson (VARZIP) Model*

Description

This function generates synthetic time series data from a Vector Autoregressive Zero-Inflated Poisson (VARZIP) model.

Usage

```
SimVARZIP(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 by (k * p). Each k by 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 by k.

Details

The SimVARZIP() function generates synthetic time series data from a Vector Autoregressive (VAR) with Zero-Inflated Poisson (ZIP) model for the first observed variable. See SimVAR() for more details on generating data for VAR(p). The SimVARZIP function goes further by using the generated values for the first variable to generate data from the ZIP model. The exponential of the values from the first variable from the original VAR(p) model are used as the intensity parameter in the Poisson distribution in the ZIP model. Data from the ZIP model are used to replace the original values for the first variable. Values for the rest of the variables are unchanged. The generated data includes a burn-in period, which is excluded before returning the results.

The steps involved in generating the 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 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 process 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 t by applying the autoregressive terms for each lag lag and each variable l.

    * Add the generated noise to the VAR time series values.
    * For the first variable, apply the Zero-Inflated Poisson (ZIP) model:
      - Compute the intensity `intensity` as the exponential of the first variable's value at time `t`.
      - Sample a random value `u` from a uniform distribution on [0, 1].
      - If `u` is less than `intensity / (1 + intensity)`, set the first variable's value to zero (inflation).
      - Otherwise, sample the first variable's value from a Poisson distribution with mean `intensity` (count process).
  - Transpose the data matrix `data` and return only the required time period after burn-in as a numeric matrix.

Value

Numeric matrix containing the simulated time series data with dimensions `k` by `(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 <- 50L
burn_in <- 10L
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(diag(3))
y <- SimVARZIP(
  time = time,
  burn_in = burn_in,
  constant = constant,
  coef = coef,
  chol_cov = chol_cov
)
head(y)
```

StdMat *Standardize Matrix*

Description

This function standardizes the given matrix by centering the columns and scaling them to have unit variance.

Usage

```
StdMat(X)
```

Arguments

X	Numeric matrix. The matrix to be standardized.

Value

Numeric matrix with standardized values.

Author(s)

Ivan Jacob Agaloos Pesigan

Examples

```
std <- StdMat(VAR)
colMeans(std)
var(std)
```

VAR *Data from the Vector Autoregressive Model*

Description

Data from the Vector Autoregressive Model

Usage

```
VAR
```

Format

A matrix with 1000 rows (time points) and 3 columns (variables) generated from the p = 2 vector autoregressive model given by

$$Y_{1_t} = 1 + 0.4Y_{1_{t-1}} + 0.0Y_{2_{t-1}} + 0.0Y_{3_{t-1}} + 0.1Y_{1_{t-2}} + 0.0Y_{2_{t-2}} + 0.0Y_{3_{t-2}} + \varepsilon_{1_t},$$

$$Y_{2_t} = 1 + 0.0Y_{1_{t-1}} + 0.5Y_{2_{t-1}} + 0.0Y_{3_{t-1}} + 0.0Y_{1_{t-2}} + 0.2Y_{2_{t-2}} + 0.0Y_{3_{t-2}} + \varepsilon_{2_t},$$

and

$$Y_{3_t} = 1 + 0.0Y_{1_{t-1}} + 0.0Y_{2_{t-1}} + 0.6Y_{3_{t-1}} + 0.0Y_{1_{t-2}} + 0.0Y_{2_{t-2}} + 0.3Y_{3_{t-2}} + \varepsilon_{3_t}$$

which simplifies to

$$Y_{1_t} = 1 + 0.4Y_{1_{t-1}} + 0.1Y_{1_{t-2}} + \varepsilon_{1_t},$$

$$Y_{2_t} = 1 + 0.5Y_{2_{t-1}} + 0.2Y_{2_{t-2}} + \varepsilon_{2_t},$$

and

$$Y_{3_t} = 1 + 0.6Y_{3_{t-1}} + 0.3Y_{3_{t-2}} + \varepsilon_{3_t}.$$

The covariance matrix of process noise is an identity matrix.

VAR_YX — *Data from the Vector Autoregressive Model (Y) and Lagged Predictors (X)*

Description

Data from the Vector Autoregressive Model (Y) and Lagged Predictors (X)

Usage

```
VAR_YX
```

Format

A list with elements Y and X where Y is equal to the `VAR` data set minus p = 2 terminal rows and `X` is a matrix of ones for the first column and lagged values of `Y` for the rest of the columns.

YX *Create Y and X Matrices*

Description

This function creates the dependent variable (Y) and predictor variable (X) matrices.

Usage

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

Arguments

data	Numeric matrix. The time series data with dimensions t by k, where t 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 t by k, where t 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 steps involved in creating the Y and X matrices are as follows:

- Determine the number of observations t 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 a column of ones and 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 dependent variable (Y) and predictor variable (X) matrices. Note that the resulting matrices will have t - p rows.

Author(s)

Ivan Jacob Agaloos Pesigan

See Also

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

Examples

```
yx <- YX(data = VAR, p = 2)
str(yx)
```

Index