Package 'hdiVAR'

May 8, 2026

Type Package

Title Statistical Inference for Noisy Vector Autoregression

Version 1.0.2

Description
The model is high-dimensional vector autoregression with measurement error, also known as linear gaussian state-space model. Provable sparse expectation-maximization algorithm is provided for the estimation of transition matrix and noise variances. Global and simultaneous testings are implemented for transition matrix with false discovery rate control. For more information, see the accompanying paper: Lyu, X., Kang, J., & Li, L. (2023). ``Statistical inference for high-dimensional vector autoregression with measurement error", Statistica Sinica.

Imports lpSolve, abind

License GPL (>= 2)

Depends R (>= 3.1)

Encoding UTF-8

RoxygenNote 7.2.3

Suggests knitr, rmarkdown

VignetteBuilder knitr

Author Xiang Lyu [aut, cre],
Jian Kang [aut],
Lexin Li [aut]

Maintainer Xiang Lyu <xianglyu.public@gmail.com>

NeedsCompilation no

Repository CRAN

Date/Publication 2023-05-14 22:00:02 UTC

Contents

CV_VARMLE 2
Estep 3
hdVARtest 4
kalman 6

Mstep 7
sEM 8
VARMLE 10

Index **12**

CV_VARMLE *cross-validation for transition matrix update in maximization step*

Description

Tune the tolerance parameter of generalized Dantzig selector and hard thresholding level via prediction error in test data.

Usage

```
CV_VARMLE(tol_seq, ht_seq, S0_train, S1_train, Y_test, is_echo = FALSE)
```

Arguments

tol_seq	vector; grid of tolerance parameter in Dantzig selector for cross-validation.
ht_seq	vector; grid of hard-thresholding levels for transition matrix estimate. To avoid hard thresholding, set ht_seq=0.
S0_train	a p by p matrix; average (over time points in training data) of conditional expectation of $x_t x_t^\top$ on $y_1, \ldots, y_T$ and parameter estimates, obtained from expectation step.
S1_train	a p by p matrix; average (over time points in training data) of conditional expectation of $x_t x_{t+1}^\top$on $y_1, \ldots, y_T$ and parameter estimates, obtained from expectation step.
Y_test	a p by T_test matrix; observations of time series in test set.
is_echo	logical; if true, display the information of CV-optimal (tol, ht).

Value

a list of CV-optimal parameters and test prediction error.

tol_min	CV-optimal tolerance parameter in Dantzig selector.
ht_min	CV-optimal hard thresholding level for the output of Dantzig selector.
test_loss	a matrix of prediction error in test data; columns match tol_seq, and rows match ht_seq.

Author(s)

Xiang Lyu, Jian Kang, Lexin Li

`Estep` *expectation step in sparse expectation-maximization algorithm*

Description

Compute conditional expectation and covariance of x_t given $y_1, \ldots, y_T$ and current parameter estimates of $A, \sigma_\eta, \sigma_\epsilon$ via kalman filter and smoothing.

Usage

```
Estep(Y,A_init,sig_eta_init,sig_epsilon_init,X_init,P_init)
```

Arguments

`Y`	observations of time series, a p by T matrix.
`A_init`	current estimate of transition matrix A.
`sig_eta_init`	current estiamte of σ_η.
`sig_epsilon_init`	current estiamte σ_ϵ.
`X_init`	current estimate of latent x_1 at the first iteration, a p-dimensional vector.
`P_init`	current covariance estimate of latent x_1 at the first iteration, a p by p matrix.

Value

a list of conditional expectations and covariances for the sequential Maximization step.

`EXtT`	a p by T matrix of column $E[x_t \vert y_1, \ldots, y_T, \hat{A}, \hat{\sigma}_\eta, \hat{\sigma}_\epsilon]$.
`EXtt`	a p by p by T tensor of first-two-mode slice $E[x_t x_t^\top \vert y_1, \ldots, y_T, \hat{A}, \hat{\sigma}_\eta, \hat{\sigma}_\epsilon]$.
`EXtt1`	a p by p by T-1 matrix of first-two-mode slice $E[x_t x_{t+1}^\top \vert y_1, \ldots, y_T, \hat{A}, \hat{\sigma}_\eta, \hat{\sigma}_\epsilon]$.

Author(s)

Xiang Lyu, Jian Kang, Lexin Li

Examples

```
p= 2; Ti=10  # dimension and time
A=diag(1,p) # transition matrix
sig_eta=sig_epsilon=0.2 # error std
Y=array(0,dim=c(p,Ti)) #observation t=1, ...., Ti
X=array(0,dim=c(p,Ti)) #latent t=1, ...., T
Ti_burnin=100 # time for burn-in to stationarity
for (t in 1:(Ti+Ti_burnin)) {
  if (t==1){
    x1=rnorm(p)
  } else if (t<=Ti_burnin) { # burn in
```

```
    x1=A%*%x1+rnorm(p,mean=0,sd=sig_eta)
  } else if (t==(Ti_burnin+1)){ # time series used for learning
    X[,t-Ti_burnin]=x1
    Y[,t-Ti_burnin]=X[,t-Ti_burnin]+rnorm(p,mean=0,sd=sig_epsilon)
  } else {
    X[,t- Ti_burnin]=A%*%X[,t-1- Ti_burnin]+rnorm(p,mean=0,sd=sig_eta)
    Y[,t- Ti_burnin]=X[,t- Ti_burnin]+rnorm(p,mean=0,sd=sig_epsilon)
  }
}

Efit=Estep(Y,A,sig_eta,sig_epsilon,x1,diag(1,p))
```

`hdVARtest` *statistical inference for transition matrix in high-dimensional vector autoregression with measurement error*

Description

Conduct global and simultaneous testing on the transition matrix.

Usage

```
hdVARtest(
  Y,
  A_est,
  sig2_eta,
  sig2_epsilon,
  global_H0 = NULL,
  global_idx = NULL,
  simul_H0 = NULL,
  simul_idx = NULL,
  FDR_levels = 0.05,
  grid_num = 2000
)
```

Arguments

`Y`	observations of time series, a p by T matrix.
`A_est`	a p by p matrix of transition matrix A estimate.
`sig2_eta`	scalar; estimate of propagation error variance σ_η^2.
`sig2_epsilon`	scalar; estimate of measurement error variance σ_ϵ^2.
`global_H0`	a p by p matrix of global null hypothesis for transition matrix A. If `global_H0=NULL`, global testing will not be conducted, and `global_idx` will not be used.

global_idx	a p by p boolean matrix. The TRUE/nonzero entry indicates the entry of interest in global hypothesis testing. If global_idx=NULL, all p*p entries are included in global testing.
simul_H0	a p by p matrix of simultaneous null hypothesis for transition matrix A. If simul_H0=NULL, simultaneous testing will not be conducted, and (simul_idx, FDR_levels, grid_num) will not be used.
simul_idx	a p by p boolean matrix. The TRUE/nonzero entry indicates the entry of interest in simultaneous hypothesis testing. If simul_idx=NULL, all p*p entries are included in simultaneous testing.
FDR_levels	a vector of FDR control levels
grid_num	scalar; the number of grids for cutoff search in FDR control.

Value

a list of testing results and gaussian test statistic matrices.

pvalue	scalar; p-value of global testing. Exist if global_H0 is not NULL.
global_test_stat	a p by p matrix of gaussian test statistic for global null hypothesis. Exist if global_H0 is not NULL.
simul_test_stat	a p by p matrix of gaussian test statistic for simultaneous null hypothesis. Exist if simul_H0 is not NUL
FDR_levels	a vector of FDR control levels. The same as input argument FDR_levels.
crt	a vector of critical values for rejecting entries in simultaneous hypothesis under corresponding FDR co
selected	a three-way tensor. The first two modes are p by p, and the third mode is for FDR control levels. Nonze

Author(s)

Xiang Lyu, Jian Kang, Lexin Li

Examples

```
p= 3; Ti=200  # dimension and time
A=diag(1,p) # transition matrix
sig_eta=sig_epsilon=0.2 # error std
Y=array(0,dim=c(p,Ti)) #observation t=1, ...., Ti
X=array(0,dim=c(p,Ti)) #latent t=1, ...., T
Ti_burnin=300 # time for burn-in to stationarity
for (t in 1:(Ti+Ti_burnin)) {
  if (t==1){
    x1=rnorm(p)
  } else if (t<=Ti_burnin) { # burn in
    x1=A%*%x1+rnorm(p,mean=0,sd=sig_eta)
  } else if (t==(Ti_burnin+1)){ # time series used for learning
    X[,t-Ti_burnin]=x1
    Y[,t-Ti_burnin]=X[,t-Ti_burnin]+rnorm(p,mean=0,sd=sig_epsilon)
  } else {
    X[,t- Ti_burnin]=A%*%X[,t-1- Ti_burnin]+rnorm(p,mean=0,sd=sig_eta)
    Y[,t- Ti_burnin]=X[,t- Ti_burnin]+rnorm(p,mean=0,sd=sig_epsilon)
  }
}

# null hypotheses are true
```

```
hdVARtest(Y,A,sig_eta^2,sig_epsilon^2,global_H0=A,global_idx=NULL,
         simul_H0=A,simul_idx=NULL,FDR_levels=c(0.05,0.1))


# null hypotheses are false
hdVARtest(Y,A,sig_eta^2,sig_epsilon^2,global_H0=matrix(0,p,p),global_idx=NULL,
          simul_H0=matrix(0,p,p),simul_idx=NULL,FDR_levels=c(0.05,0.1))
```

`kalman` *kalman filtering and smoothing for vector autoregression with measurement error*

Description

kalman filtering and smoothing for vector autoregression with measurement error

Usage

```
kalman(Y,A,sig_eta,sig_epsilon,X_init=NULL,P_init=NULL)
```

Arguments

`Y`	observations of time series, a p by T matrix.
`A`	current estimate of transition matrix.
`sig_eta`	current estiamte of σ_η.
`sig_epsilon`	current estiamte σ_ϵ.
`X_init`	inital estimate of latent x_1 at the first iteration, a p-dimensional vector.
`P_init`	inital covariance estimate of latent x_1 at the first iteration, a p by p matrix.

Value

a list of conditional expectations and covariances of x_t's.

Author(s)

Xiang Lyu, Jian Kang, Lexin Li

`Mstep` *maximization step of sparse expectation-maximization algorithm for updating error standard deviations*

Description

Update $\sigma_\eta, \sigma_\epsilon$ based on estimate of A and conditional expecation and covariance from expectation step.

Usage

```
Mstep(Y,A,EXtT,EXtt,EXtt1,is_MLE=FALSE)
```

Arguments

`Y`	observations of time series, a p by T matrix.
`A`	current estimate of transition matrix A. If `is_MLE=TRUE`, use naive MLE of transition matrix, by conditional expecation and covariance from expecation step, to update error standard deviations.
`EXtT`	a p by T matrix of column $E[x_t \vert y_1, \ldots, y_T, \hat{A}, \hat{\sigma}_\eta, \hat{\sigma}_\epsilon]$ from expectation step.
`EXtt`	a p by p by T tensor of first-two-mode slice $E[x_t x_t^\top \vert y_1, \ldots, y_T, \hat{A}, \hat{\sigma}_\eta, \hat{\sigma}_\epsilon]$ from expectation step.
`EXtt1`	a p by p by T-1 matrix of first-two-mode slice $E[x_t x_{t+1}^\top \vert y_1, \ldots, y_T, \hat{A}, \hat{\sigma}_\eta, \hat{\sigma}_\epsilon]$ from expectation step.
`is_MLE`	logic; if true, use naive MLE of transition matrix, by conditional expecation and covariance from expecation step, to update error variances. Otherwise, use input argument `A`.

Value

a list of estimates of error standard deviations.

`sig_eta`	estimate of σ_η.
`sig_epsilon`	estimate of σ_ϵ.
`A`	naive MLE of transition matrix A by conditional expecation and covariance from expecation step. Exist if `is_`

Author(s)

Xiang Lyu, Jian Kang, Lexin Li

Examples

```
p= 2; Ti=10  # dimension and time
A=diag(1,p) # transition matrix
sig_eta=sig_epsilon=0.2 # error std
Y=array(0,dim=c(p,Ti)) #observation t=1, ...., Ti
X=array(0,dim=c(p,Ti)) #latent t=1, ...., T
Ti_burnin=100 # time for burn-in to stationarity
for (t in 1:(Ti+Ti_burnin)) {
  if (t==1){
    x1=rnorm(p)
  } else if (t<=Ti_burnin) { # burn in
    x1=A%*%x1+rnorm(p,mean=0,sd=sig_eta)
  } else if (t==(Ti_burnin+1)){ # time series used for learning
    X[,t-Ti_burnin]=x1
    Y[,t-Ti_burnin]=X[,t-Ti_burnin]+rnorm(p,mean=0,sd=sig_epsilon)
  } else {
    X[,t- Ti_burnin]=A%*%X[,t-1- Ti_burnin]+rnorm(p,mean=0,sd=sig_eta)
    Y[,t- Ti_burnin]=X[,t- Ti_burnin]+rnorm(p,mean=0,sd=sig_epsilon)
  }
}

# expectation step
Efit=Estep(Y,A,sig_eta,sig_epsilon,x1,diag(1,p))
EXtT=Efit[["EXtT"]]
EXtt=Efit[["EXtt"]]
EXtt1=Efit[["EXtt1"]]
# maximization step for error standard deviations
Mfit=Mstep(Y,A,EXtT,EXtt,EXtt1)
```

---

sEM *sparse expectation-maximization algorithm for high-dimensional vector autoregression with measurement error*

Description

Alteranting between expectation step (by kalman filter and smoothing) and maximization step (by generalized Dantzig selector for transiton matrix) to estimate transtion matrix and error variances.

Usage

```
sEM(
  Y,
  A_init,
  sig2_eta_init,
  sig2_epsilon_init,
  Ti_train = NULL,
```

```
  Ti_gap = NULL,
  tol_seq = NULL,
  ht_seq = 0,
  is_cv = TRUE,
  thres = 0.001,
  count_vanish = 1,
  n_em = NULL,
  is_echo = FALSE,
  is_sparse = TRUE
)
```

Arguments

`Y`	observations of time series, a p by T matrix.
`A_init`	a p by p matrix as initial value of transition matrix A estimate.
`sig2_eta_init`	scalar; initial value of propagation error variance σ_η^2 estimate in latent signal process.
`sig2_epsilon_init`	scalar; initial value of measurement error variance σ_ϵ^2 estimate in observation process.
`Ti_train`	scalar; the number of time points in training data in cross-validation.
`Ti_gap`	scalar; the number of time points between test data and train data in cross-validation.
`tol_seq`	vector; grid of tolerance parameter in Dantzig selector for cross-validation. If `is_cv=FALSE`, use the first element.
`ht_seq`	vector; grid of hard-thresholding levels for transition matrix estimate. If `is_cv=FALSE`, use the first element. To avoid hard thresholding, set `ht_seq=0`.
`is_cv`	logical; if true, run cross-validation to tune Dantzig selector tolerance parameter each sparse EM iteration.
`thres`	scalar; if the difference between updates of two consecutive iterations is less that `thres`, record one hit. The algorithm is terminated due to vanishing updates if hit times accumulate up to `count_vanish`. If `thres=NULL`, the algorithm will not be terminated due to vanishing updates, but too many iterations instead.
`count_vanish`	scalar; if the difference between updates of two consecutive iterations is less that `thres` up to `count_vanish` times, the algorithm is terminated due to vanishing updates.
`n_em`	scalar; the maximal allowed number of EM iterations, otherwise the algorithm is terminated due to too many iterations. If `n_em=NULL`, the algorithm will not be terminated due to too many iterations, but vanishing updates instead.
`is_echo`	logical; if true, display the information of CV-optimal (tol, ht) each iteration, and of algorithm termination.
`is_sparse`	logical; if false, use standard EM algorithm, and arguments for cross-validation are not needed.

Value

a list of parameter estimates.

A_est	estimate of transition matrix A.
sig2_eta_hat	estimate of propagation error variance σ_η^2.
sig2_epsilon_hat	estimate of measurement error variance σ_ϵ^2.
iter_err	the difference between updates of two consecutive iterations.
iter_err_ratio	the difference ratio (over the previous estimate) between updates of two consecutive iterations.

Author(s)

Xiang Lyu, Jian Kang, Lexin Li

Examples

```
p= 3; Ti=20  # dimension and time
A=diag(1,p) # transition matrix
sig_eta=sig_epsilon=0.2 # error std
Y=array(0,dim=c(p,Ti)) #observation t=1, ...., Ti
X=array(0,dim=c(p,Ti)) #latent t=1, ...., T
Ti_burnin=30 # time for burn-in to stationarity
for (t in 1:(Ti+Ti_burnin)) {
  if (t==1){
    x1=rnorm(p)
  } else if (t<=Ti_burnin) { # burn in
    x1=A%*%x1+rnorm(p,mean=0,sd=sig_eta)
  } else if (t==(Ti_burnin+1)){ # time series used for learning
    X[,t-Ti_burnin]=x1
    Y[,t-Ti_burnin]=X[,t-Ti_burnin]+rnorm(p,mean=0,sd=sig_epsilon)
  } else {
    X[,t- Ti_burnin]=A%*%X[,t-1- Ti_burnin]+rnorm(p,mean=0,sd=sig_eta)
    Y[,t- Ti_burnin]=X[,t- Ti_burnin]+rnorm(p,mean=0,sd=sig_epsilon)
  }
}

sEM_fit=sEM(Y,diag(0.5,p),0.1,0.1,Ti*0.5,Ti*0.2,c(0.01,0.1))
```

VARMLE *generalized Dantzig selector for transition matrix update in maximization step*

Description

Sparse estimation of transtion matrix in vector autoregression given conditional autocovariance matrices.

Usage

```
VARMLE(S0, S1, tol)
```

Arguments

`S0`	a p by p matrix; average (over time points) of conditional expectation of $x_t x_t^\top$ on $y_1, \ldots, y_T$ and parameter estimates, obtained from expectation step.
`S1`	a p by p matrix; average (over time points) of conditional expectation of $x_t x_{t+1}^\top$ on $y_1, \ldots, y_T$ and parameter estimates, obtained from expectation step.
`tol`	tolerance parameter in Dantzig selector.

Value

Sparse estimate of transition matrix by Dantzig selector.

Author(s)

Xiang Lyu, Jian Kang, Lexin Li

Index

CV_VARMLE, 2

Estep, 3

hdVARtest, 4

kalman, 6

Mstep, 7

sEM, 8

VARMLE, 10