Package: VARDetect (via r-universe)

October 14, 2024

Type Package

Title Multiple Change Point Detection in Structural VAR Models

Version 0.1.8

Author Yue Bai [aut, cre], Peiliang Bai [aut], Abolfazl Safikhani [aut], George Michailidis [aut]

Maintainer Yue Bai <baiyue69@gmail.com>

Description Implementations of Thresholded Block Segmentation Scheme (TBSS) and Low-rank plus Sparse Two Step Procedure (LSTSP) algorithms for detecting multiple changes in structural VAR models. The package aims to address the problem of change point detection in piece-wise stationary VAR models, under different settings regarding the structure of their transition matrices (autoregressive dynamics); specifically, the following cases are included: (i) (weakly) sparse, (ii) structured sparse, and (iii) low rank plus sparse. It includes multiple algorithms and related extensions from Safikhani and Shojaie (2020) <doi:10.1080/01621459.2020.1770097> and Bai, Safikhani and Michailidis (2020) <doi:10.1109/TSP.2020.2993145>.

License GPL-2

LazyData true

Encoding UTF-8

Depends R (>= 3.5.0)

Imports stats, MTS, igraph, pracma, graphics, mvtnorm, sparsevar, lattice, Rcpp (>= 1.0.7)

LinkingTo Rcpp, RcppArmadillo

Suggests knitr, rmarkdown

RoxygenNote 7.3.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2024-06-15 18:20:06 UTC

Contents

`detection_check` *Function for detection performance check*

Description

Function for detection performance check

Usage

```
detection_check(pts.final, brk, nob, critval = 5)
```

Arguments

`pts.final`	a list of estimated change points
`brk`	the true change points
`nob`	length of time series
`critval`	critical value for selection rate. Default value is 5. Specifically, to compute the selection rate, a selected break point is counted as a "success" for the j-th true break point, t_j, if it falls in the interval $[t_j - (t_j - t_{j-1})/critval, t_j + (t_{j+1} - t_j)/critval]$, $j = 1, \ldots, m_0$.

Value

a matrix of detection summary results, including the absolute error, selection rate and relative location. The absolute error of the locations of the estimated break points is defined as $error_j = |\tilde{t}_j^f - t_j|, j = 1, \ldots, m_0$.

Examples

```
# an example of 10 replicates result
set.seed(1)
nob <- 1000
brk <- c(333, 666, nob+1)
cp.list <- vector('list', 10)
for(i in 1:10){
    cp.list[[i]] <-  brk[1:2] + sample(c(-50:50),1)
}
# some replicate fails to detect all the change point
cp.list[[2]] <- cp.list[[2]][1]
cp.list[4] <- list(NULL)      # setting 4'th element to NULL.
# some replicate overestimate the number of change point
cp.list[[3]] <- c(cp.list[[3]], 800)
cp.list
res <- detection_check(cp.list, brk, nob, critval = 5)
res
# use a stricter critical value
res <- detection_check(cp.list, brk, nob, critval = 10)
res
```

---

eeg *EEG signal data*

Description

EEG signal data

Usage

```
data(eeg)
```

Format

An dataframe of EEG signal data

Examples

```
data(eeg)
head(eeg)
```

`eval_func` *Evaluation function, return the performance of simulation results*

Description

Evaluation function, return the performance of simulation results

Usage

```
eval_func(true_mats, est_mats)
```

Arguments

`true_mats`	a list of true matrices for all segments, the length of list equals to the true number of segments
`est_mats`	a list of estimated matrices for all simulation replications, for each element, it is a list of numeric matrices, representing the estimated matrices for segments

Value

A list, containing the results for all measurements

sensitivity A numeric vector, containing all the results for sensitivity over all replications

specificity A numeric vector, including all the results for specificity over all replications

accuracy A numeric vector, the results for accuracy over all replications

mcc A numeric vector, the results for Matthew's correlation coefficients over all replications

false_reps An integer vector, recording all the replications which falsely detects the change points, over-detect or under-detect

Examples

```
true_mats <- vector('list', 2)
true_mats[[1]] <- matrix(c(1, 0, 0.5, 0.8), 2, 2, byrow = TRUE)
true_mats[[2]] <- matrix(c(0, 0, 0, 0.75), 2, 2, byrow = TRUE)
est_mats <- vector('list', 5)
for(i in 1:5){
    est_mats[[i]] <- vector('list', 2)
   est_mats[[i]][[1]] <- matrix(sample(c(0, 1, 2), size = 4, replace = TRUE), 2, 2, byrow = TRUE)
   est_mats[[i]][[2]] <- matrix(sample(c(0, 1), size = 4, replace = TRUE), 2, 2, byrow = TRUE)
}
perf_eval <- eval_func(true_mats, est_mats)
```

hausdorff_check *Function for Hausdorff distance computation*

Description

The function includes two Hausdorff distance. The first one is hausdorff_true_est ($d(A_n, \tilde{A}_n^f)$): for each estimated change point, we find the closest true CP and compute the distance, then take the maximum of distances. The second one is hausdorff_est_true($d(\tilde{A}_n^f, A_n)$): for each true change point, find the closest estimated change point and compute the distance, then take the maximum of distances.

Usage

```
hausdorff_check(pts.final, brk)
```

Arguments

pts.final	a list of estimated change points
brk	the true change points

Value

Hausdorff distance summary results, including mean, standard deviation and median.

Examples

```
## an example of 10 replicates result
set.seed(1)
nob <- 1000
brk <- c(333, 666, nob+1)
cp.list <- vector('list', 10)
for(i in 1:10){
    cp.list[[i]] <-  brk[1:2] + sample(c(-50:50),1)
}
# some replicate fails to detect all the change point
cp.list[[2]] <- cp.list[[2]][1]
cp.list[4] <- list(NULL)      # setting 4'th element to NULL.
# some replicate overestimate the number of change point
cp.list[[3]] <- c(cp.list[[3]], 800)
cp.list
res <- hausdorff_check(cp.list, brk)
res
```

`lag_selection` *Select the lag of the VAR model using total BIC method*

Description

Select the lag of the VAR model (if the lag is unknown) using BIC method for total segments

Usage

```
lag_selection(
  data,
  method = c("sparse", "group sparse", "fLS"),
  group.case = c("columnwise", "rowwise"),
  group.index = NULL,
  lambda.1.cv = NULL,
  lambda.2.cv = NULL,
  mu = NULL,
  block.size = NULL,
  blocks = NULL,
  use.BIC = TRUE,
  an.grid = NULL,
  threshold = NULL,
  lag_candidates,
  verbose = FALSE
)
```

Arguments

`data`	input data matrix, each column represents the time series component
`method`	method is sparse, group sparse and fixed lowrank plus sparse
`group.case`	two different types of group sparse, column-wise and row-wise, respectively.
`group.index`	specify group sparse index. Default is NULL.
`lambda.1.cv`	tuning parameter lambda_1 for fused lasso
`lambda.2.cv`	tuning parameter lambda_2 for fused lasso
`mu`	tuning parameter for low rank component, only available when method is set to "fLS".
`block.size`	the block size
`blocks`	the blocks
`use.BIC`	use BIC for k-means part
`an.grid`	a vector of an for grid searching.
`threshold`	a numeric argument, give the threshold for estimated model parameter matrices. Default is NULL.
`lag_candidates`	potential lag selection set
`verbose`	A Boolean argument, if TRUE, it provides detailed information. Default is FALSE

Value

selected lag for VAR series

select_lag An integer no less than 1 represents the selected lag of time series.

Examples

```
nob <- 1000; p <- 15
brk <- c(floor(nob / 2), nob + 1)
m <- length(brk)
q.t <- 2 # the lag of VAR model for simulation
signals <- c(-0.8, 0.6, 0.4)
try <- simu_var(method = "sparse", nob = nob, k = p, brk = brk,
                signals = signals, lags_vector = c(1, 2),
                sp_pattern = "off-diagonal")
data <- try$series; data <- as.matrix(data)

# Apply lag selection to determine the lag for the given time series
lag_candi <- c(1, 2, 3, 4)
select_lag <- lag_selection(data = data,
                            method = "sparse", lag_candidates = lag_candi)
print(select_lag)
```

---

lstsp *Main function for the low rank plus sparse structure VAR model*

Description

Main function for the low-rank plus sparse structure VAR model

Usage

```
lstsp(
  data,
  lambda.1 = NULL,
  mu.1 = NULL,
  lambda.1.seq = NULL,
  mu.1.seq = NULL,
  lambda.2 = NULL,
  mu.2 = NULL,
  lambda.3 = NULL,
  mu.3 = NULL,
  alpha_L = 0.25,
  omega = NULL,
  h = NULL,
  step.size = NULL,
  tol = 1e-04,
```

```
  niter = 100,
  backtracking = TRUE,
  skip = 5,
  cv = FALSE,
  nfold = NULL,
  verbose = FALSE
)
```

Arguments

data	A n by p dataset matrix
lambda.1	tuning parameter for sparse component for the first step
mu.1	tuning parameter for low rank component for the first step
lambda.1.seq	a sequence of lambda to the left segment for cross-validation, it's not mandatory to provide
mu.1.seq	a sequence of mu to the left segment, low rank component tuning parameter
lambda.2	tuning parameter for sparse for the second step
mu.2	tuning parameter for low rank for the second step
lambda.3	tuning parameter for estimating sparse components
mu.3	tuning parameter for estimating low rank components
alpha_L	a positive numeric value, indicating the restricted space of low rank component, default is 0.25
omega	tuning parameter for information criterion, the larger of omega, the fewer final selected change points
h	window size of the first rolling window step
step.size	rolling step
tol	tolerance for the convergence in the second screening step, indicates when to stop
niter	the number of iterations required for FISTA algorithm
backtracking	A boolean argument to indicate use backtrack to FISTA model
skip	The number of observations need to skip near the boundaries
cv	A boolean argument, indicates whether the user will apply cross validation to select tuning parameter, default is FALSE
nfold	An positive integer, the number of folds for cross validation
verbose	If is TRUE, then it will print all information about current step.

Value

A list object including

data the original dataset

q the time lag for the time series, in this case, it is 1

cp Final estimated change points

sparse_mats Final estimated sparse components

lowrank_mats Final estimated low rank components

est_phi Final estimated model parameter, equals to sum of low rank and sparse components

time Running time for the LSTSP algorithm

Examples

```
nob <- 100
p <- 15
brk <- c(50, nob+1)
rank <- c(1, 3)
signals <- c(-0.7, 0.8)
singular_vals <- c(1, 0.75, 0.5)
info_ratio <- rep(0.35, 2)
try <- simu_var(method = "LS", nob = nob, k = p, lags = 1, brk = brk,
                sigma = as.matrix(diag(p)), signals = signals,
                rank = rank, singular_vals = singular_vals, info_ratio = info_ratio,
                sp_pattern = "off-diagonal", spectral_radius = 0.9)
data <- try$series

lambda1 = lambda2 = lambda3 <- c(2.5, 2.5)
mu1 = mu2 = mu3 <- c(15, 15)
fit <- lstsp(data, lambda.1 = lambda1, mu.1 = mu1,
             lambda.2 = lambda2, mu.2 = mu2,
             lambda.3 = lambda3, mu.3 = mu3, alpha_L = 0.25,
             step.size = 5, niter = 20, skip = 5,
             cv = FALSE, verbose = FALSE)
summary(fit)
plot(fit, data, display = "cp")
plot(fit, data, display = "param")
```

---

plot.VARDetect.result *Plotting the output from VARDetect.result class*

Description

Plotting method for S3 object of class `VARDetect.result`

Usage

```
## S3 method for class 'VARDetect.result'
plot(
  x,
  display = c("cp", "param", "granger", "density"),
  threshold = 0.1,
  layout = c("circle", "star", "nicely"),
  ...
)
```

Arguments

x	a VARDetect.result object
display	a character string, indicates the object the user wants to plot; possible values are "cp" input time series together with the estimated change points "param" estimated model parameters "granger" present the model parameters through Granger causal networks "density" plot the sparsity levels across all segments
threshold	a positive numeric value, indicates the threshold to present the entries in the sparse matrices
layout	a character string, indicating the layout of the Granger network
...	not in use

Value

A plot for change points or a series of plots for Granger causal networks for estimated model parameters

Examples

```
nob <- 1000
p <- 15
brk <- c(floor(nob / 3), floor(2 * nob / 3), nob + 1)
m <- length(brk)
q.t <- 1
try <- simu_var('sparse',nob=nob,k=p,lags=q.t,brk=brk,sp_pattern="off-diagonal",seed = 1)
data <- try$series
data <- as.matrix(data)
fit <- tbss(data, method = "sparse", q = q.t)
plot(fit, display = "cp")
plot(fit, display = "param")
plot(fit, display = "granger", threshold = 0.2, layout = "nicely")
plot(fit, display = "density", threshold = 0.2)
```

---

plot_density *Function to plot the sparsity levels for estimated model parameters*

Description

A function to plot lineplot for sparsity levels of estimated model parameters

Usage

```
plot_density(est_mats, threshold = 0.1)
```

Arguments

est_mats	A list of numeric matrices, the length of list equals to the number of estimated segments
threshold	A numeric value, set as a threshold, the function only counts the non-zeros with absolute magnitudes larger than threshold

Value

A plot for sparsity density across over all estimated segments

Examples

```
set.seed(1)
est_mats <- list(matrix(rnorm(400, 0, 2), 20, 20), matrix(rnorm(400), 20, 20))
plot_density(est_mats, threshold = 0.25)
```

---

plot_granger *Function to plot Granger causality networks*

Description

A function to plot Granger causal network for each segment via estimated sparse component. Note that if it has multiple lags, it only provides the first order Granger causality plot.

Usage

```
plot_granger(est_mats, threshold = 0.1, layout)
```

Arguments

est_mats	A list of numeric sparse matrices, indicating the estimated sparse components for each segment
threshold	A numeric positive value, used to determine the threshold to present the edges
layout	A character string, indicates the layout for the igraph plot argument

Value

A series of plots of Granger networks of VAR model parameters

Examples

```
set.seed(1)
est_mats <- list(matrix(rnorm(400, 0, 1), 20, 20))
plot_granger(est_mats, threshold = 2, layout = "circle")
plot_granger(est_mats, threshold = 2, layout = "star")
plot_granger(est_mats, threshold = 2, layout = "nicely")
```

plot_matrix *Plot the AR coefficient matrix*

Description

Plot the AR coefficient matrix

Usage

```
plot_matrix(phi, p)
```

Arguments

phi	combined coefficient matrices for all lags
p	number of segments times number of lags

Value

a plot of AR coefficient matrix

Examples

```
nob <- 4 * 10^3
p <- 15
brk <- c(floor(nob / 3), floor(2 * nob / 3), nob + 1)
m0 <- length(brk) - 1
q.t <- 2
m <- m0 + 1
sp_density <- rep(0.05, m*q.t) #sparsity level (5%)
try <- simu_var("sparse", nob = nob, k = p, lags = q.t, brk = brk,
                sp_pattern = "random", sp_density = sp_density)
print(plot_matrix(do.call("cbind", try$model_param), m * q.t))
```

print.VARDetect.result *Function to print the change points estimated by VARDetect*

Description

Print the estimated change points of class `VARDetect.result`

Usage

```
## S3 method for class 'VARDetect.result'
print(x, ...)
```

Arguments

x	a VARDetect.result class object
...	not in use

Value

Print the estimated change points

Examples

```
nob <- 1000
p <- 15
brk <- c(floor(nob / 3), floor(2 * nob / 3), nob + 1)
m <- length(brk)
q.t <- 1
try <- simu_var('sparse',nob=nob,k=p,lags=q.t,brk=brk,sp_pattern="off-diagonal",seed=1)
data <- try$series
data <- as.matrix(data)
fit <- tbss(data, method = "sparse", q = q.t)
print(fit)
```

---

simu_lstsp *Function to deploy simulation with LSTSP algorithm*

Description

A function to generate simulation with LSTSP algorithm

Usage

```
simu_lstsp(
  nreps,
  simu_method = c("LS"),
  nob,
  k,
  lags = 1,
  lags_vector = NULL,
  brk,
  sigma,
  skip = 50,
  group_mats = NULL,
  group_type = c("columnwise", "rowwise"),
  group_index = NULL,
  sparse_mats = NULL,
  sp_density = NULL,
  signals = NULL,
  rank = NULL,
```

```
  info_ratio = NULL,
  sp_pattern = c("off-diagonal", "diagoanl", "random"),
  singular_vals = NULL,
  spectral_radius = 0.9,
  alpha_L = 0.25,
  lambda.1 = NULL,
  mu.1 = NULL,
  lambda.1.seq = NULL,
  mu.1.seq = NULL,
  lambda.2,
  mu.2,
  lambda.3,
  mu.3,
  omega = NULL,
  h = NULL,
  step.size = NULL,
  tol = 1e-04,
  niter = 100,
  backtracking = TRUE,
  rolling.skip = 5,
  cv = FALSE,
  nfold = NULL,
  verbose = FALSE
)
```

Arguments

nreps	A positive integer, indicating the number of simulation replications
simu_method	the structure of time series: only available for "LS"
nob	sample size
k	dimension of transition matrix
lags	lags of VAR time series. Default is 1.
lags_vector	a vector of lags of VAR time series for each segment
brk	a vector of break points with (nob+1) as the last element
sigma	the variance matrix for error term
skip	an argument to control the leading data points to obtain a stationary time series
group_mats	transition matrix for group sparse case
group_type	type for group lasso: "columnwise", "rowwise". Default is "columnwise".
group_index	group index for group lasso.
sparse_mats	transition matrix for sparse case
sp_density	if we choose random pattern, we should provide the sparsity density for each segment
signals	manually setting signal for each segment (including sign)
rank	if we choose method is low rank plus sparse, we need to provide the ranks for each segment

info_ratio	the information ratio leverages the signal strength from low rank and sparse components
sp_pattern	a choice of the pattern of sparse component: diagonal, 1-off diagonal, random, custom
singular_vals	singular values for the low rank components
spectral_radius	to ensure the time series is piecewise stationary.
alpha_L	a positive numeric value, indicating the restricted space of low rank component, default is 0.25
lambda.1	tuning parameter for sparse component for the first step
mu.1	tuning parameter for low rank component for the first step
lambda.1.seq	a sequence of lambda to the left segment for cross-validation, it's not mandatory to provide
mu.1.seq	a sequence of mu to the left segment, low rank component tuning parameter
lambda.2	tuning parameter for sparse for the second step
mu.2	tuning parameter for low rank for the second step
lambda.3	tuning parameter for estimating sparse components
mu.3	tuning parameter for estimating low rank components
omega	tuning parameter for information criterion, the larger of omega, the fewer final selected change points
h	window size of the first rolling window step
step.size	rolling step
tol	tolerance for the convergence in the second screening step, indicates when to stop
niter	the number of iterations required for FISTA algorithm
backtracking	A boolean argument to indicate use backtrack to FISTA model
rolling.skip	The number of observations need to skip near the boundaries
cv	A boolean argument, indicates whether the user will apply cross validation to select tuning parameter, default is FALSE
nfold	An positive integer, the number of folds for cross validation
verbose	If is TRUE, then it will print all information about current step.

Value

A S3 object of class `VARDetect.simu.result`, containing the following entries:

sizes A 2-d numeric vector, indicating the size of time series data

true_lag True time lags for the process, here is fixed to be 1.

true_lagvector A vector recording the time lags for different segments, not available under this model setting, here is fixed to be NULL

true_cp True change points for simulation, a numeric vector

true_sparse A list of numeric matrices, indicating the true sparse components for all segments

true_lowrank A list of numeric matrices, indicating the true low rank components for all segments

est_cps A list of estimated change points, including all replications

est_lag A numeric value, estimated time lags, which is user specified

est_lagvector A vector for estimated time lags, not available for this model, set as NULL.

est_sparse_mats A list of estimated sparse components for all replications

est_lowrank_mats A list of estimated low rank components for all replications

est_phi_mats A list of estimated model parameters, transition matrices for VAR model

running_times A numeric vector, containing all running times

Examples

```
nob <- 100
p <- 15
brk <- c(50, nob+1)
rank <- c(1, 3)
signals <- c(-0.7, 0.8)
singular_vals <- c(1, 0.75, 0.5)
info_ratio <- rep(0.35, 2)
lambda1 = lambda2 = lambda3 <- c(2.5, 2.5)
mu1 = mu2 = mu3 <- c(15, 15)
try_simu <- simu_lstsp(nreps = 3, simu_method = "LS", nob = nob, k = p,
                       brk = brk, sigma = diag(p), signals = signals,
                       rank = rank, singular_vals = singular_vals,
                       info_ratio = info_ratio, sp_pattern = "off-diagonal",
                       spectral_radius = 0.9, lambda.1 = lambda1, mu.1 = mu1,
                       lambda.2 = lambda2, mu.2 = mu2, lambda.3 = lambda3,
                       mu.3 = mu3, step.size = 5, niter = 20, rolling.skip = 5,
                       cv = FALSE, verbose = TRUE)
summary(try_simu, critical = 5)
```

---

simu_tbss *Simulation function for TBSS algorithm*

Description

Function for deploying simulation using TBSS algorithm

Usage

```
simu_tbss(
  nreps,
  simu_method = c("sparse", "group sparse", "fLS"),
  nob,
  k,
```

```
  lags = 1,
  lags_vector = NULL,
  brk,
  sigma,
  skip = 50,
  group_mats = NULL,
  group_type = c("columnwise", "rowwise"),
  group_index = NULL,
  sparse_mats = NULL,
  sp_density = NULL,
  signals = NULL,
  rank = NULL,
  info_ratio = NULL,
  sp_pattern = c("off-diagonal", "diagoanl", "random"),
  singular_vals = NULL,
  spectral_radius = 0.9,
  est_method = c("sparse", "group sparse", "fLS"),
  q = 1,
  tol = 0.01,
  lambda.1.cv = NULL,
  lambda.2.cv = NULL,
  mu = NULL,
  group.index = NULL,
  group.case = c("columnwise", "rowwise"),
  max.iteration = 100,
  refit = FALSE,
  block.size = NULL,
  blocks = NULL,
  use.BIC = TRUE,
  an.grid = NULL,
  verbose = FALSE
)
```

Arguments

nreps	A numeric integer number, indicates the number of simulation replications
simu_method	the structure of time series: "sparse","group sparse", and "fLS"
nob	sample size
k	dimension of transition matrix
lags	lags of VAR time series. Default is 1.
lags_vector	a vector of lags of VAR time series for each segment
brk	a vector of break points with (nob+1) as the last element
sigma	the variance matrix for error term
skip	an argument to control the leading data points to obtain a stationary time series
group_mats	transition matrix for group sparse case
group_type	type for group lasso: "columnwise", "rowwise". Default is "columnwise".

group_index	group index for group lasso.
sparse_mats	transition matrix for sparse case
sp_density	if we choose random pattern, we should provide the sparsity density for each segment
signals	manually setting signal for each segment (including sign)
rank	if we choose method is low rank plus sparse, we need to provide the ranks for each segment
info_ratio	the information ratio leverages the signal strength from low rank and sparse components
sp_pattern	a choice of the pattern of sparse component: diagonal, 1-off diagonal, random, custom
singular_vals	singular values for the low rank components
spectral_radius	to ensure the time series is piecewise stationary.
est_method	method: sparse, group sparse, and fixed low rank plus sparse. Default is sparse
q	the AR order
tol	tolerance for the fused lasso
lambda.1.cv	tuning parameter lambda_1 for fused lasso
lambda.2.cv	tuning parameter lambda_2 for fused lasso
mu	tuning parameter for low rank component, only available when method is set to "fLS"
group.index	group index for group sparse case
group.case	group sparse pattern: column, row.
max.iteration	max number of iteration for the fused lasso
refit	logical; if TRUE, refit the VAR model for parameter estimation. Default is FALSE.
block.size	the block size
blocks	the blocks
use.BIC	use BIC for k-means part
an.grid	a vector of an for grid searching
verbose	a Boolean argument; if TRUE, function provides detailed information. Default is FALSE

Value

A S3 object of class, named `VARDetect.simu.result`

est_cps A list of estimated change points, including all replications

est_sparse_mats A list of estimated sparse components for all replications

est_lowrank_mats A list of estimated low rank components for all replications

est_phi_mats A list of estimated model parameters, transition matrices for VAR model

running_times A numeric vector, containing all running times

Examples

```
nob <- 4000; p <- 15
brk <- c(floor(nob / 3), floor(2 * nob / 3), nob + 1)
m <- length(brk); q.t <- 1
sp_density <- rep(0.05, m * q.t)
signals <- c(-0.6, 0.6, -0.6)
try_simu <- simu_tbss(nreps = 3, simu_method = "sparse", nob = nob,
                      k = p, lags = q.t, brk = brk, sigma = diag(p),
                      signals = signals, sp_density = sp_density,
                      sp_pattern = "random", est_method = "sparse", q = q.t,
                      refit = TRUE)
```

---

simu_var *Generate VAR(p) model data with break points*

Description

This function is used for generate simulated time series

Usage

```
simu_var(
  method = c("sparse", "group sparse", "fLS", "LS"),
  nob = 300,
  k = 20,
  lags = 1,
  lags_vector = NULL,
  brk,
  sigma = NULL,
  skip = 50,
  spectral_radius = 0.98,
  seed = NULL,
  sp_density = NULL,
  group_mats = NULL,
  group_index = NULL,
  group_type = c("columnwise", "rowwise"),
  sparse_mats = NULL,
  sp_pattern = c("off-diagonal", "diagonal", "random"),
  rank = NULL,
  info_ratio = NULL,
  signals = NULL,
  singular_vals = NULL
)
```

Arguments

method	the structure of time series: "sparse","group sparse", "fLS", "LS"
nob	sample size
k	dimension of transition matrix
lags	lags of VAR time series. Default is 1.
lags_vector	a vector of lags of VAR time series for each segment
brk	a vector of break points with (nob+1) as the last element
sigma	the variance matrix for error term
skip	an argument to control the leading data points to obtain a stationary time series
spectral_radius	to ensure the time series is piecewise stationary.
seed	an argument to control the random seed. Default seed is 1.
sp_density	if we choose random pattern, we should provide the sparsity density for each segment
group_mats	transition matrix for group sparse case
group_index	group index for group lasso.
group_type	type for group lasso: "columnwise", "rowwise". Default is "columnwise".
sparse_mats	transition matrix for sparse case
sp_pattern	a choice of the pattern of sparse component: diagonal, 1-off diagonal, random, custom
rank	if we choose method is low rank plus sparse, we need to provide the ranks for each segment
info_ratio	the information ratio leverages the signal strength from low rank and sparse components
signals	manually setting signal for each segment (including sign)
singular_vals	singular values for the low rank components

Value

A list object, which contains the followings

series matrix of timeseries data

noises matrix of noise term data

sparse_mats list of sparse matrix in the transition matrix

lowrank_mats list of low-rank matrix in the transition matrix

Examples

```
nob <- (10^3 * 4) # number of time points
p <- 15 # number of time series components
brk <- c(floor(nob / 3), floor(2 * nob / 3), nob + 1)
m0 <- length(brk) - 1 # number of break points
q.t <- 2 # the true AR order
m <- m0 + 1 # number of segments
sp_density <- rep(0.05, m * q.t) # sparsity level (5%)
try <- simu_var("sparse", nob = nob, k = p, lags = q.t, brk = brk,
                sp_pattern = "random", sp_density = sp_density)
print(plot_matrix(do.call("cbind", try$model_param), m * q.t))
```

---

summary.VARDetect.result

Function to summarize the change points estimated by VARDetect

---

Description

Summary method for objects of class `VARDetect.result`

Usage

```
## S3 method for class 'VARDetect.result'
summary(object, threshold = 0.1, ...)
```

Arguments

`object`	a `VARDetect.result` object
`threshold`	A numeric positive value, used to determine the threshold of nonzero entries
`...`	not in use

Value

A series of summary, including the estimated change points, running time

Examples

```
nob <- 1000
p <- 15
brk <- c(floor(nob / 3), floor(2 * nob / 3), nob + 1)
m <- length(brk)
q.t <- 1
try <- simu_var('sparse',nob=nob,k=p,lags=q.t,brk=brk,sp_pattern="off-diagonal",seed=1)
data <- try$series
data <- as.matrix(data)
fit <- tbss(data, method = "sparse", q = q.t)
summary(fit)
```

summary.VARDetect.simu.result
A function to summarize the results for simulation

Description

A function to summarize the results for simulation class `VARDetect.simu.result`

Usage

```
## S3 method for class 'VARDetect.simu.result'
summary(object, critical = 5, ...)
```

Arguments

`object`	A S3 object of class `VARDetect.simu.result`
`critical`	A positive integer, set as the critical value defined in selection rate, to control the range of success, default is 5
`...`	not in use

Value

A series of summary, including the selection rate, Hausdorff distance, and statistical measurements, running times

Examples

```
nob <- 4000; p <- 15
brk <- c(floor(nob / 3), floor(2 * nob / 3), nob + 1)
m <- length(brk); q.t <- 1
sp_density <- rep(0.05, m * q.t)
signals <- c(-0.6, 0.6, -0.6)
try_simu <- simu_tbss(nreps = 3, simu_method = "sparse", nob = nob,
                      k = p, lags = q.t, brk = brk, sigma = diag(p),
                      signals = signals, sp_density = sp_density,
                      sp_pattern = "random", est_method = "sparse",
                      q = q.t, refit = TRUE)
summary(try_simu, critical = 5)
```

tbss *Block segmentation scheme (BSS).*

Description

Perform the block segmentation scheme (BSS) algorithm to detect the structural breaks in large scale high-dimensional non-stationary VAR models.

Usage

```
tbss(
  data,
  method = c("sparse", "group sparse", "fLS"),
  group.case = c("columnwise", "rowwise"),
  group.index = NULL,
  lambda.1.cv = NULL,
  lambda.2.cv = NULL,
  mu = NULL,
  q = 1,
  max.iteration = 50,
  tol = 10^(-2),
  block.size = NULL,
  blocks = NULL,
  refit = FALSE,
  use.BIC = TRUE,
  an.grid = NULL,
  verbose = FALSE
)
```

Arguments

data	input data matrix, with each column representing the time series component
method	method: sparse, group sparse, and fixed low rank plus sparse. Default is sparse
group.case	group sparse pattern: column, row.
group.index	group index for group sparse case
lambda.1.cv	tuning parameter lambda_1 for fused lasso
lambda.2.cv	tuning parameter lambda_2 for fused lasso
mu	tuning parameter for low rank component, only available when method is set to "fLS"
q	the VAR lag
max.iteration	max number of iteration for the Fused lasso
tol	tolerance for the fused lasso
block.size	the block size
blocks	the blocks

refit	logical; if TRUE, refit the VAR model for parameter estimation. Default is FALSE.
use.BIC	use BIC for k-means part
an.grid	a vector of an for grid searching
verbose	a Boolean argument to determine whether provide detailed outputs for each step. Default is FALSE

Value

S3 object of class VARDetect.result, which contains the followings

data the original dataset

q the time lag user specified, a numeric value

cp final estimated change points, a numeric vector

sparse_mats estimated sparse components for each segment, a list of numeric matrices

lowrank_mats estimated low rank components for each segment, a list of numeric matrices

est_phi estimated final model parameters, the summation of the sparse and the low rank components

time computation time for each step

Examples

```
#### sparse VAR model
nob <- (10^3) # number of time points
p <- 15; # number of time series components
brk <- c(floor(nob/3),floor(2*nob/3),nob+1); # true break points with nob+1 as the last element
m0 <- length(brk) -1; # number of break points
q.t <- 1; # the true AR order
m <- m0+1 #number of segments
try<-simu_var('sparse',nob=nob,k=p,lags=q.t,brk = brk,sp_pattern="off-diagonal",seed=1)
data <- try$series
data <- as.matrix(data)
#run the bss method
fit <- tbss(data, method = "sparse", q = q.t)
print(fit)
summary(fit)
plot(fit, data, display = "cp")
plot(fit, data, display = "param")


###### Example for fixed low rank plus sparse structure VAR model
nob <- 300
p <- 15
brk <- c(floor(nob/3), floor(2*nob/3), nob+1)
m <- length(brk)
q.t <- 1
signals <- c(-0.7, 0.7, -0.7)
rank <- c(2, 2, 2)
singular_vals <- c(1, 0.75)
```

```
info_ratio <- rep(0.35, 3)
try <- simu_var(method = "fLS", nob = nob, k = p, lags = 1, brk = brk,
                sigma = as.matrix(diag(p)), signals = signals, seed=1,
                rank = rank, singular_vals = singular_vals, info_ratio = info_ratio,
                sp_pattern = "off-diagonal", spectral_radius = 0.9)
data <- try$series
data <- as.matrix(data)
fit <- tbss(data, method = "fLS", mu = 150)
print(fit)
summary(fit)
plot(fit, data, display = "cp")
plot(fit, data, display = "param")
```

---

weekly *weekly stock price data*

Description

weekly stock price data

Usage

```
data(weekly)
```

Format

An dataframe of weekly stock price data

Examples

```
data(weekly)
head(weekly)
```

Index