Package: ssaBSS (via r-universe)

August 24, 2024

Type Package

Title Stationary Subspace Analysis

Version 0.1.1

Date 2022-12-01

Maintainer Markus Matilainen <markus.matilainen@outlook.com>

Depends tsBSS (>= 0.5.3), ICtest (>= 0.3-4), JADE (>= 2.0-2), BSSprep, ggplot2

Imports xts, zoo

Description Stationary subspace analysis (SSA) is a blind source separation (BSS) variant where stationary components are separated from non-stationary components. Several SSA methods for multivariate time series are provided here (Flumian et al. (2021); Hara et al. (2010) <doi:10.1007/978-3-642-17537-4_52>) along with functions to simulate time series with time-varying variance and autocovariance (Patilea and Raissi(2014) <doi:10.1080/01621459.2014.884504>).

License GPL (>= 2)

NeedsCompilation no

Author Markus Matilainen [cre, aut] (<https://orcid.org/0000-0002-5597-2670>), Lea Flumian [aut], Klaus Nordhausen [aut] (<https://orcid.org/0000-0002-3758-8501>), Sara Taskinen [aut] (<https://orcid.org/0000-0001-9470-7258>)

Repository CRAN

Date/Publication 2022-12-01 16:40:05 UTC

Contents

`ssaBSS-package` *Stationary Subspace Analysis*

Description

Stationary subspace analysis (SSA) is a blind source separation (BSS) variant where stationary components are separated from non-stationary components. Several SSA methods for multivariate time series are provided here (Flumian et al. (2021); Hara et al. (2010) <doi:10.1007/978-3-642-17537-4_52>) along with functions to simulate time series with time-varying variance and autocovariance (Patilea and Raïssi(2014) <doi:10.1080/01621459.2014.884504>).

Details

Package:	ssaBSS
Type:	Package
Version:	0.1.1
Date:	2022-12-01
License:	GPL (>= 2)

This package contains functions for identifying different types of nonstationarity

- `SSAsir` – SIR type function for mean non-stationarity identification
- `SSAsave` – SAVE type function for variance non-stationarity identification
- `SSAcor` – Function for identifying changes in autocorrelation
- `ASSA` – ASSA: Analytic SSA for identification of nonstationarity in mean and variance.
- `SSAcomb` – Combination of `SSAsir`, `SSAsave`, and `SSAcor` using joint diagonalization

The package also contains function `rtvvar` to simulate a time series with time-varying variance (TV-VAR), and function `rtvAR1` to simulate a time series with time-varying autocovariance (TV-AR1).

Author(s)

Markus Matilainen, Léa Flumian, Klaus Nordhausen, Sara Taskinen

Maintainer: Markus Matilainen <markus.matilainen@outlook.com>

References

Flumian L., Matilainen M., Nordhausen K. and Taskinen S. (2021) *Stationary subspace analysis based on second-order statistics*. Submitted. Available on arXiv: https://arxiv.org/abs/2103.06148

Hara S., Kawahara Y., Washio T. and von Bünau P. (2010). *Stationary Subspace Analysis as a Generalized Eigenvalue Problem*, Neural Information Processing. Theory and Algorithms, Part I, pp. 422-429.

Patilea V. and Raïssi H. (2014) *Testing Second-Order Dynamics for Autoregressive Processes in Presence of Time-Varying Variance*, Journal of the American Statistical Association, 109 (507), 1099-1111.

`ASSA` *ASSA Method for Non-stationary Identification*

Description

ASSA (Analytic Stationary Subspace Analysis) method for identifying non-stationary components of mean and variance.

Usage

```
ASSA(X, ...)

## Default S3 method:
ASSA(X, K, n.cuts = NULL, ...)
## S3 method for class 'ts'
ASSA(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. Missing values are not allowed.
`K`	Number of intervals the time series is split into.
`n.cuts`	A K+1 vector of values that correspond to the breaks which are used for splitting the data. Default is intervals of equal length.
`...`	Further arguments to be passed to or from methods.

Details

Assume that a p-variate $\mathbf{Y}$ with T observations is whitened, i.e. $\mathbf{Y} = \mathbf{S}^{-1/2}(\mathbf{X}_t - \frac{1}{T}\sum_{t=1}^{T}\mathbf{X}_t)$, where $\mathbf{S}$ is the sample covariance matrix of $\mathbf{X}$.

The values of $\mathbf{Y}$ are then split into K disjoint intervals T_i. Algorithm first calculates matrix

$$\mathbf{M} = \frac{1}{T}\sum_{i=1}^{K}\left(\mathbf{m}_{T_i}\mathbf{m}_{T_i}^T + \frac{1}{2}\mathbf{S}_{T_i}\mathbf{S}_{T_i}^T\right) - \frac{1}{2}\mathbf{I},$$

where K is the number of breakpoints, $\mathbf{I}$ is an identity matrix, and $\mathbf{m}_{T_i}$ is the average of values of $\mathbf{Y}$ and $\mathbf{S}_{T_i}$ is the sample variance of values of $\mathbf{Y}$ which belong to a disjoint interval T_i.

The algorithm finds an orthogonal matrix $\mathbf{U}$ via eigendecomposition

$$\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{U}^T.$$

The final unmixing matrix is then $\mathbf{W} = \mathbf{U}\mathbf{S}^{-1/2}$. The first k rows of $\mathbf{U}$ are the eigenvectors corresponding to the non-zero eigenvalues and the rest correspond to the zero eigenvalues. In the same way, the first k rows of $\mathbf{W}$ project the observed time series to the subspace of non-stationary components, and the last $p-k$ rows to the subspace of stationary components.

Value

A list of class 'ssabss', inheriting from class 'bss', containing the following components:

`W`	The estimated unmixing matrix.
`S`	The estimated sources as time series object standardized to have mean 0 and unit variances.
`M`	Used separation matrix.
`K`	Number of intervals the time series is split into.
`D`	Eigenvalues of M.
`MU`	The mean vector of `X`.
`n.cut`	Used K+1 vector of values that correspond to the breaks which are used for splitting the data.
`method`	Name of the method ("ASSA"), to be used in e.g. screeplot.

Author(s)

Markus Matilainen, Klaus Nordhausen

References

Hara S., Kawahara Y., Washio T. and von Bünau P. (2010). *Stationary Subspace Analysis as a Generalized Eigenvalue Problem*, Neural Information Processing. Theory and Algorithms, Part I, pp. 422-429.

See Also

JADE

Examples

```
n <- 5000
A <- rorth(4)

z1 <- arima.sim(n, model = list(ar = 0.7)) + rep(c(-1.52, 1.38), c(floor(n*0.5),
        n - floor(n*0.5)))
z2 <- rtvvar(n, alpha = 0.1, beta = 1)
```

```
z3 <- arima.sim(n, model = list(ma = c(0.72, 0.24)))
z4 <- arima.sim(n, model = list(ar = c(0.34, 0.27, 0.18)))

Z <- cbind(z1, z2, z3, z4)
X <- as.ts(tcrossprod(Z, A)) # An mts object

res <- ASSA(X, K = 6)
screeplot(res, type = "lines") # Two non-zero eigenvalues

# Plotting the components as an mts object
plot(res) # The first two are nonstationary
```

---

rtvAR1 *Simulation of Time Series with Time-varying Autocovariance*

Description

Simulating time-varying variance based on TV-AR1 model

Usage

```
rtvAR1(n, sigma = 0.93)
```

Arguments

n	Length of the time series
sigma	Parameter σ^2 in TV-AR1, i.e. the variance. Default is 0.93.

Details

Time varying autoregressive processes of order 1 (TV-AR1) is

$$x_t = a_t x_{t-1} + \epsilon_t,$$

with $x_0 = 0$, ϵ_t is iid $N(0, \sigma^2)$ and $a_t = 0.5\cos(2\pi t/T)$.

Value

The simulated series as a ts object.

Author(s)

Sara Taskinen, Markus Matilainen

References

Patilea V. and Raïssi H. (2014) *Testing Second-Order Dynamics for Autoregressive Processes in Presence of Time-Varying Variance*, Journal of the American Statistical Association, 109 (507), 1099-1111.

Examples

```
n <- 5000
X <- rtvAR1(n, sigma = 0.93)
plot(X)
```

---

rtvvar *Simulation of Time Series with Time-varying Variance*

Description

Simulating time-varying variance based on TV-VAR model

Usage

```
rtvvar(n, alpha, beta = 1, simple = FALSE)
```

Arguments

`n`	Length of the time series
`alpha`	Parameter α in TV-VAR
`beta`	Parameter β in TV-VAR. Default is 1.
`simple`	A logical vector indicating whether h_t is considered as its own process, or just t/T. Default is FALSE.

Details

Time varying variance (TV-VAR) process x_t with parameters α and β is of the form

$$x_t = \tilde{h}_t \epsilon_t,$$

where, if `simple = FALSE`,

$$\tilde{h}_t^2 = h_t^2 + \alpha x_{t-1}^2,$$

where ϵ are iid $N(0,1)$, $x_0 = 0$ and $h_t = 10 - 10\sin(\beta\pi t/T + \pi/6)(1 + t/T)$,

and if `simple = TRUE`,

$$\tilde{h}_t = t/T.$$

Value

The simulated series as a `ts` object.

Author(s)

Sara Taskinen, Markus Matilainen

References

Patilea V. and Raïssi H. (2014) *Testing Second-Order Dynamics for Autoregressive Processes in Presence of Time-Varying Variance*, Journal of the American Statistical Association, 109 (507), 1099-1111.

Examples

```
n <- 5000
X <- rtvvar(n, alpha = 0.2, beta = 0.5, simple = FALSE)
plot(X)
```

---

ssabss *Class: ssabss*

Description

Class 'ssabss' (blind source separation in stationary subspace analysis) with methods plot, screeplot (prints a screeplot of an object of class 'ssabss') and ggscreeplot (prints a screeplot of an object of class 'ssabss' using package ggplot2).

The class 'ssabss' also inherits methods from the class 'bss' in package JADE: for extracting the components (bss.components), for plotting the components (plot.bss), for printing (print.bss), and for extracting the coefficients (coef.bss).

Usage

```
## S3 method for class 'ssabss'
plot(x, ...)

## S3 method for class 'ssabss'
screeplot(x, type = c("lines", "barplot"), xlab = "Number of components",
          ylab = NULL, main = paste("Screeplot for", x$method),
          pointsize = 4, breaks = 1:length(x$D), color = "red", ...)

## S3 method for class 'ssabss'
ggscreeplot(x, type = c("lines", "barplot"), xlab = "Number of components",
            ylab = NULL, main = paste("Screeplot for", x$method),
            pointsize = 4, breaks = 1:length(x$D), color = "red", ...)
```

Arguments

x	An object of class 'ssabss'.
type	Type of screeplot. Choices are "lines" (default) and "barplot".
xlab	Label for x-axis. Default is "Number of components".
ylab	Label for y-axis. Default is "Sum of pseudo eigenvalues" if method is SSAcomb and "Eigenvalues" otherwise.

`main`	Title of the plot. Default is "Screeplot for ...", where ... denotes for the name of the method used.
`pointsize`	Size of the points in the plot (for `type = "lines"` only). Default is 4.
`breaks`	Breaks and labels for the x-axis. Default is from 1 to the number of series by 1.
`color`	Color of the line (if `type = "lines"`) or bar (if `type = "barplot"`). Default is red.
`...`	Further arguments to be passed to or from methods.

Details

A screeplot can be used to determine the number of interesting components. For `SSAcomb` it plots the sum of pseudo eigenvalues and for other methods it plots the eigenvalues.

Author(s)

Markus Matilainen

See Also

`ASSA`, `SSAsir`, `SSAsave`, `SSAcor`, `SSAcomb`, `JADE`, `ggplot2`

`SSAcomb` *Combination Main SSA Methods*

Description

SSAcomb method for identification for non-stationarity in mean, variance and covariance structure.

Usage

```
SSAcomb(X, ...)

## Default S3 method:
SSAcomb(X, K, n.cuts = NULL, tau = 1, eps = 1e-6, maxiter = 2000, ...)
## S3 method for class 'ts'
SSAcomb(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. Missing values are not allowed.
`K`	Number of intervals the time series is split into.
`n.cuts`	A K+1 vector of values that correspond to the breaks which are used for splitting the data. Default is intervals of equal length.
`tau`	The lag as a scalar or a vector. Default is 1.
`eps`	Convergence tolerance.
`maxiter`	The maximum number of iterations.
`...`	Further arguments to be passed to or from methods.

Details

Assume that a p-variate $\mathbf{Y}$ with T observations is whitened, i.e. $\mathbf{Y} = \mathbf{S}^{-1/2}(\mathbf{X}_t - \frac{1}{T}\sum_{t=1}^{T}\mathbf{X}_t)$, where $\mathbf{S}$ is the sample covariance matrix of $\mathbf{X}$.

The values of $\mathbf{Y}$ are then split into K disjoint intervals T_i. For all lags $j = 1, ..., ntau$, algorithm first calculates the $\mathbf{M}$ matrices from SSAsir (matrix $\mathbf{M}_1$), SSAsave (matrix $\mathbf{M}_2$) and SSAcor (matrices $\mathbf{M}_{j+2}$).

The algorithm finds an orthogonal matrix $\mathbf{U}$ by maximizing

$$\sum_{i=1}^{ntau+2} ||\mathrm{diag}(\mathbf{U}\mathbf{M}_i\mathbf{U}')||^2.$$

The final unmixing matrix is then $\mathbf{W} = \mathbf{U}\mathbf{S}^{-1/2}$.

Then the pseudo eigenvalues $\mathbf{D}_i = \mathrm{diag}(\mathbf{U}\mathbf{M}_i\mathbf{U}') = \mathrm{diag}(d_{i,1}, \dots, d_{i,p})$ are obtained and the value of $d_{i,j}$ tells if the jth component is nonstationary with respect to $\mathbf{M}_i$.

Value

A list of class 'ssabss', inheriting from class 'bss', containing the following components:

`W`	The estimated unmixing matrix.
`S`	The estimated sources as time series object standardized to have mean 0 and unit variances.
`R`	Used M-matrices as an array.
`K`	Number of intervals the time series is split into.
`D`	The sums of pseudo eigenvalues.
`DTable`	The peudo eigenvalues of size *ntau + 2* to see which type of nonstationarity there exists in each component.
`MU`	The mean vector of `X`.
`n.cut`	Used K+1 vector of values that correspond to the breaks which are used for splitting the data.
`k`	The used lag.
`method`	Name of the method ("SSAcomb"), to be used in e.g. screeplot.

Author(s)

Markus Matilainen, Klaus Nordhausen

References

Flumian L., Matilainen M., Nordhausen K. and Taskinen S. (2021) *Stationary subspace analysis based on second-order statistics*. Submitted. Available on arXiv: https://arxiv.org/abs/2103.06148

See Also

`JADE` `frjd`

Examples

```
n <- 10000
A <- rorth(6)

z1 <- arima.sim(n, model = list(ar = 0.7)) + rep(c(-1.52, 1.38),
        c(floor(n*0.5), n - floor(n*0.5)))
z2 <- rtvAR1(n)
z3 <- rtvvar(n, alpha = 0.2, beta = 0.5)
z4 <- arima.sim(n, model = list(ma = c(0.72, 0.24), ar = c(0.14, 0.45)))
z5 <- arima.sim(n, model = list(ma = c(0.34)))
z6 <- arima.sim(n, model = list(ma = c(0.72, 0.15)))

Z <- cbind(z1, z2, z3, z4, z5, z6)
library(xts)
X <- tcrossprod(Z, A)
X <- xts(X, order.by = as.Date(1:n)) # An xts object

res <- SSAcomb(X, K = 12, tau = 1)
ggscreeplot(res, type = "lines") # Three non-zero eigenvalues
res$DTable # Components have different kind of nonstationarities

# Plotting the components as an xts object
plot(res, multi.panel = TRUE) # The first three are nonstationary
```

SSAcor — *Identification of Non-stationarity in the Covariance Structure*

Description

SSAcor method for identifying non-stationarity in the covariance structure.

Usage

```
SSAcor(X, ...)

## Default S3 method:
SSAcor(X, K, n.cuts = NULL, tau = 1, eps = 1e-6, maxiter = 2000, ...)
## S3 method for class 'ts'
SSAcor(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. Missing values are not allowed.
`K`	Number of intervals the time series is split into.

`n.cuts`	A K+1 vector of values that correspond to the breaks which are used for splitting the data. Default is intervals of equal length.
`tau`	The lag as a scalar or a vector. Default is 1.
`eps`	Convergence tolerance.
`maxiter`	The maximum number of iterations.
`...`	Further arguments to be passed to or from methods.

Details

Assume that a p-variate $\mathbf{Y}$ with T observations is whitened, i.e. $\mathbf{Y} = \mathbf{S}^{-1/2}(\mathbf{X}_t - \frac{1}{T}\sum_{t=1}^{T}\mathbf{X}_t)$, where $\mathbf{S}$ is the sample covariance matrix of $\mathbf{X}$.

The values of $\mathbf{Y}$ are then split into K disjoint intervals T_i. For all lags $j = 1, ..., ntau$, algorithm first calculates matrices

$$\mathbf{M_j} = \sum_{i=1}^{K} \frac{T_i}{T}(\mathbf{S}_{j,T} - \mathbf{S}_{j,T_i})(\mathbf{S}_{j,T} - \mathbf{S}_{j,T_i})^T,$$

where K is the number of breakpoints, $\mathbf{S}_{J,T}$ is the global sample covariance for lag j, and $\mathbf{S}_{\tau,T_i}$ is the sample covariance of values of $\mathbf{Y}$ which belong to a disjoint interval T_i.

The algorithm finds an orthogonal matrix $\mathbf{U}$ by maximizing

$$\sum_{j=1}^{ntau} ||\mathrm{diag}(\mathbf{U}\mathbf{M}_j\mathbf{U}')||^2.$$

The final unmixing matrix is then $\mathbf{W} = \mathbf{U}\mathbf{S}^{-1/2}$. Then the pseudo eigenvalues $\mathbf{D}_i = \mathrm{diag}(\mathbf{U}\mathbf{M}_i\mathbf{U}') = \mathrm{diag}(d_{i,1}, \ldots, d_{i,p})$ are obtained and the value of $d_{i,j}$ tells if the jth component is nonstationary with respect to $\mathbf{M}_i$. The first k rows of $\mathbf{W}$ project the observed time series to the subspace of components with non-stationary covariance, and the last $p - k$ rows to the subspace of components with stationary covariance.

Value

A list of class 'ssabss', inheriting from class 'bss', containing the following components:

`W`	The estimated unmixing matrix.
`S`	The estimated sources as time series object standardized to have mean 0 and unit variances.
`M`	Used separation matrix.
`K`	Number of intervals the time series is split into.
`D`	The sums of pseudo eigenvalues.
`DTable`	The peudo eigenvalues of size *ntau**p* to see which type of nonstationarity there exists in each component.
`MU`	The mean vector of `X`.
`n.cut`	Used K+1 vector of values that correspond to the breaks which are used for splitting the data.
`k`	The used lag.
`method`	Name of the method ("SSAcor"), to be used in e.g. screeplot.

Author(s)

Markus Matilainen, Klaus Nordhausen

References

Flumian L., Matilainen M., Nordhausen K. and Taskinen S. (2021) *Stationary subspace analysis based on second-order statistics*. Submitted. Available on arXiv: https://arxiv.org/abs/2103.06148

See Also

JADE

Examples

```
n <- 5000
A <- rorth(4)

z1 <- rtvAR1(n)
z2a <- arima.sim(floor(n/3), model = list(ar = c(0.5),
        innov = c(rnorm(floor(n/3), 0, 1))))
z2b <- arima.sim(floor(n/3), model = list(ar = c(0.2),
        innov = c(rnorm(floor(n/3), 0, 1.28))))
z2c <- arima.sim(n - 2*floor(n/3), model = list(ar = c(0.8),
        innov = c(rnorm(n - 2*floor(n/3), 0, 0.48))))
z2 <- c(z2a, z2b, z2c)
z3 <- arima.sim(n, model = list(ma = c(0.72, 0.24), ar = c(0.14, 0.45)))
z4 <- arima.sim(n, model = list(ar = c(0.34, 0.27, 0.18)))

Z <- cbind(z1, z2, z3, z4)
library(zoo)
X <- as.zoo(tcrossprod(Z, A)) # A zoo object

res <- SSAcor(X, K = 6, tau = 1)
ggscreeplot(res, type = "barplot", color = "gray") # Two non-zero eigenvalues

# Plotting the components as a zoo object
plot(res) # The first two are nonstationary in autocovariance
```

SSAsave *Identification of Non-stationarity in Variance*

Description

SSAsave method for identifying non-stationarity in variance

Usage

```
SSAsave(X, ...)

## Default S3 method:
SSAsave(X, K, n.cuts = NULL, ...)
## S3 method for class 'ts'
SSAsave(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. Missing values are not allowed.
`K`	Number of intervals the time series is split into.
`n.cuts`	A K+1 vector of values that correspond to the breaks which are used for splitting the data. Default is intervals of equal length.
`...`	Further arguments to be passed to or from methods.

Details

Assume that a p-variate $\mathbf{Y}$ with T observations is whitened, i.e. $\mathbf{Y} = \mathbf{S}^{-1/2}(\mathbf{X}_t - \frac{1}{T}\sum_{t=1}^{T}\mathbf{X}_t)$, where $\mathbf{S}$ is the sample covariance matrix of $\mathbf{X}$.

The values of $\mathbf{Y}$ are then split into K disjoint intervals T_i. Algorithm first calculates matrix

$$\mathbf{M} = \sum_{i=1}^{K} \frac{T_i}{T}(\mathbf{I} - \mathbf{S}_{T_i})(\mathbf{I} - \mathbf{S}_{T_i})^T,$$

where K is the number of breakpoints, $\mathbf{I}$ is an identity matrix, and $\mathbf{S}_{T_i}$ is the sample variance of values of $\mathbf{Y}$ which belong to a disjoint interval T_i.

The algorithm finds an orthogonal matrix $\mathbf{U}$ via eigendecomposition

$$\mathbf{M} = \mathbf{UDU}^T.$$

The final unmixing matrix is then $\mathbf{W} = \mathbf{US}^{-1/2}$. The first k rows of $\mathbf{U}$ are the eigenvectors corresponding to the non-zero eigenvalues and the rest correspond to the zero eigenvalues. In the same way, the first k rows of $\mathbf{W}$ project the observed time series to the subspace of components with non-stationary variance, and the last $p-k$ rows to the subspace of components with stationary variance.

Value

A list of class 'ssabss', inheriting from class 'bss', containing the following components:

`W`	The estimated unmixing matrix.
`S`	The estimated sources as time series object standardized to have mean 0 and unit variances.
`M`	Used separation matrix.
`K`	Number of intervals the time series is split into.

D	Eigenvalues of M.
MU	The mean vector of X.
n.cut	Used K+1 vector of values that correspond to the breaks which are used for splitting the data.
method	Name of the method ("SSAsave"), to be used in e.g. screeplot.

Author(s)

Markus Matilainen, Klaus Nordhausen

References

Flumian L., Matilainen M., Nordhausen K. and Taskinen S. (2021) *Stationary subspace analysis based on second-order statistics*. Submitted. Available on arXiv: https://arxiv.org/abs/2103.06148

See Also

JADE

Examples

```
n <- 5000
A <- rorth(4)

z1 <- rtvvar(n, alpha = 0.2, beta = 0.5)
z2 <- rtvvar(n, alpha = 0.1, beta = 1)
z3 <- arima.sim(n, model = list(ma = c(0.72, 0.24)))
z4 <- arima.sim(n, model = list(ar = c(0.34, 0.27, 0.18)))

Z <- cbind(z1, z2, z3, z4)
X <- as.ts(tcrossprod(Z, A)) # An mts object

res <- SSAsave(X, K = 6)
res$D # Two non-zero eigenvalues
screeplot(res, type = "lines") # This can also be seen in screeplot
ggscreeplot(res, type = "lines") # ggplot version of screeplot

# Plotting the components as an mts object
plot(res) # The first two are nonstationary in variance
```

SSAsir *Identification of Non-stationarity in Mean*

Description

SSAsir method for identifying non-stationarity in mean.

Usage

```
SSAsir(X, ...)

## Default S3 method:
SSAsir(X, K, n.cuts = NULL, ...)
## S3 method for class 'ts'
SSAsir(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. Missing values are not allowed.
`K`	Number of intervals the time series is split into.
`n.cuts`	A K+1 vector of values that correspond to the breaks which are used for splitting the data. Default is intervals of equal length.
`...`	Further arguments to be passed to or from methods.

Details

Assume that a p-variate $\mathbf{Y}$ with T observations is whitened, i.e. $\mathbf{Y} = \mathbf{S}^{-1/2}(\mathbf{X}_t - \frac{1}{T}\sum_{t=1}^{T}\mathbf{X}_t)$, where $\mathbf{S}$ is the sample covariance matrix of $\mathbf{X}$.

The values of $\mathbf{Y}$ are then split into K disjoint intervals T_i. Algorithm first calculates matrix

$$\mathbf{M} = \sum_{i=1}^{K} \frac{T_i}{T}(\mathbf{m}_{T_i}\mathbf{m}_{T_i}^T),$$

where K is the number of breakpoints, and $\mathbf{m}_{T_i}$ is the average of values of $\mathbf{Y}$ which belong to a disjoint interval T_i.

The algorithm finds an orthogonal matrix $\mathbf{U}$ via eigendecomposition

$$\mathbf{M} = \mathbf{U}\mathbf{D}\mathbf{U}^T.$$

The final unmixing matrix is then $\mathbf{W} = \mathbf{U}\mathbf{S}^{-1/2}$. The first k rows of $\mathbf{U}$ are the eigenvectors corresponding to the non-zero eigenvalues and the rest correspond to the zero eigenvalues. In the same way, the first k rows of $\mathbf{W}$ project the observed time series to the subspace of components with non-stationary mean, and the last $p - k$ rows to the subspace of components with stationary mean.

Value

A list of class 'ssabss', inheriting from class 'bss', containing the following components:

`W`	The estimated unmixing matrix.
`S`	The estimated sources as time series object standardized to have mean 0 and unit variances.
`M`	Used separation matrix.
`K`	Number of intervals the time series is split into.

D	Eigenvalues of M.
MU	The mean vector of X.
n.cut	Used K+1 vector of values that correspond to the breaks which are used for splitting the data.
method	Name of the method ("SSAsir"), to be used in e.g. screeplot.

Author(s)

Markus Matilainen, Klaus Nordhausen

References

Flumian L., Matilainen M., Nordhausen K. and Taskinen S. (2021) *Stationary subspace analysis based on second-order statistics*. Submitted. Available on arXiv: https://arxiv.org/abs/2103.06148

See Also

JADE

Examples

```
n <- 5000
A <- rorth(4)

z1 <- arima.sim(n, model = list(ar = 0.7)) + rep(c(-1.52, 1.38),
        c(floor(n*0.5), n - floor(n*0.5)))
z2 <- arima.sim(n, model = list(ar = 0.5)) + rep(c(-0.75, 0.84, -0.45),
        c(floor(n/3), floor(n/3), n - 2*floor(n/3)))
z3 <- arima.sim(n, model = list(ma = 0.72))
z4 <- arima.sim(n, model = list(ma = c(0.34)))

Z <- cbind(z1, z2, z3, z4)
X <- tcrossprod(Z, A)

res <- SSAsir(X, K = 6)
res$D # Two non-zero eigenvalues
screeplot(res, type = "lines") # This can also be seen in screeplot

# Plotting the components
plot(res) # The first two are nonstationary in mean
```

Index