

# Package ‘tsBSS’

November 17, 2017

**Type** Package

**Title** Blind Source Separation and Supervised Dimension Reduction for Time Series

**Version** 0.4

**Date** 2017-11-17

**Author** Markus Matilainen, Christophe Croux, Jari Miettinen, Klaus Nordhausen, Hannu Oja, Sara Taskinen

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

**Depends** ICtest, JADE

**Imports** Rcpp (>= 0.11.0), forecast

**LinkingTo** Rcpp, RcppArmadillo

**Suggests** stochvol

**Description** Different estimates are provided to solve the blind source separation problem for multivariate time series with stochastic volatility (Matilainen, Nordhausen and Oja (2015) <doi:10.1016/j.spl.2015.04.033>); Matilainen, Miettinen, Nordhausen, Oja and Taskinen (2017) <doi:10.17713/ajs.v46i3-4.671>) and supervised dimension reduction problem for multivariate time series (Matilainen, Croux, Nordhausen and Oja (2017) <doi:10.1016/j.ecosta.2017.04.002>).

**License** GPL (>= 2)

**LazyData** true

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2017-11-17 10:52:21 UTC

## R topics documented:

tsBSS-package	2
coef.summary.tssdr	3
components.summary.tssdr	4
components.tssdr	5
FixNA	5

gFOBI . . . . .	8
gJADE . . . . .	11
gSOBI . . . . .	13
lbttest . . . . .	16
plot.summary.tssdr . . . . .	18
plot.tssdr . . . . .	19
print.bssvol . . . . .	19
print.lbttest . . . . .	20
print.summary.tssdr . . . . .	20
print.tssdr . . . . .	21
PVC . . . . .	22
summary.tssdr . . . . .	24
tssdr . . . . .	26
vSOBI . . . . .	29
WeeklyReturnsData . . . . .	31

<b>Index</b>	<b>34</b>
--------------	-----------

---

tsBSS-package	<i>Blind Source Separation and Supervised Dimension Reduction for Time Series</i>
---------------	---

---

## Description

Different estimates are provided to solve the blind source separation problem for multivariate time series with stochastic volatility (Matilainen, Nordhausen and Oja (2015) <doi:10.1016/j.spl.2015.04.033>); Matilainen, Miettinen, Nordhausen, Oja and Taskinen (2017) <doi:10.17713/ajs.v46i3-4.671>) and supervised dimension reduction problem for multivariate time series (Matilainen, Croux, Nordhausen and Oja (2017) <doi:10.1016/j.ecosta.2017.04.002>).

## Details

Package: tsBSS  
 Type: Package  
 Version: 0.4  
 Date: 2017-11-17  
 License: GPL (>= 2)

This package contains functions for the blind source separation (BSS) problem for multivariate time series. The methods are designed for time series with stochastic volatility, such as GARCH and SV models. The main functions of the package for the BSS problem are

- [FixNA](#) Function to solve the BSS problem. Algorithm is an alternative to [vSOBI](#) algorithm to accommodate stochastic volatility.
- [gFOBI](#) Function to solve the BSS problem. Algorithm is a generalization of [FOBI](#) designed for time series with stochastic volatility.

- [gJADE](#) Function to solve the BSS problem. Algorithm is a generalization of [JADE](#) designed for time series with stochastic volatility.
- [vSOBI](#) Function to solve the BSS problem. Algorithm is a variant of [SOBI](#) algorithm and an alternative to [FixNA](#) to accommodate stochastic volatility.

The main function of the package for the supervised dimension reduction is

- [tssdr](#) Function for supervised dimension reduction for multivariate time series. Includes methods TSIR, TSAVE and TSSH.

Methods for ARMA models, such as AMUSE and SOBI, and some non-stationary BSS methods for time series are implemented in the [JADE-package](#). See function [dr](#) for methods for supervised dimension reduction for iid observations.

The package also contains a dataset [WeeklyReturnsData](#), that has logarithmic Returns of Exchange Rates of 7 Currencies Against US Dollar.

### Author(s)

Markus Matilainen, Christophe Croux, Jari Miettinen, Klaus Nordhausen, Hannu Oja, Sara Taskinen

Maintainer: Markus Matilainen <markus.matilainen@utu.fi>

### References

Matilainen, M., Nordhausen, K. and Oja, H. (2015), *New independent component analysis tools for time series*, Statistics & Probability Letters, 105, 80–87.

Matilainen, M., Miettinen, J., Nordhausen, K., Oja, H. and Taskinen, S. (2017), *On Independent Component Analysis with Stochastic Volatility Models*, Austrian Journal of Statistics, 46(3–4), 57–66.

Matilainen M., Croux C., Nordhausen K. & Oja H. (2017), *Supervised Dimension Reduction for Multivariate Time Series*, Econometrics and Statistics, 4, 57–69.

Matilainen M., Croux C., Nordhausen K. & Oja H. *Sliced Average Variance Estimation for multivariate Time Series*. Submitted.

Shi, Z., Jiang, Z. and Zhou, F. (2009), *Blind Source Separation with Nonlinear Autocorrelation and Non-Gaussianity*, Journal of Computational and Applied Mathematics, 223(1): 908–915.

Miettinen, M., Matilainen, M., Nordhausen, K. and Taskinen, S. (2017), *Extracting Conditionally Heteroscedastic Components Using ICA*, Submitted.

---

coef.summary.tssdr      *The coefficients of a summary.tssdr object*

---

### Description

Extracts the estimated signal separation matrix from an object of class summary.tssdr.

**Usage**

```
## S3 method for class 'summary.tssdr'  
coef(object, ...)
```

**Arguments**

object            An object of class summary.tssdr.  
...               Further arguments to be passed to or from methods.

**Author(s)**

Markus Matilainen

**See Also**

[summary.tssdr](#)

---

components.summary.tssdr

*Directions of an object of class summary.tssdr*

---

**Description**

Gives the directions of an object of class summary.tssdr.

**Usage**

```
## S3 method for class 'summary.tssdr'  
components(x, ...)
```

**Arguments**

x                 An object of class summary.tssdr.  
...               Further arguments to be passed to or from methods.

**Author(s)**

Markus Matilainen

**See Also**

[summary.tssdr](#)

---

components.tssdr      *Directions of an object of class tssdr*

---

### Description

Gives directions of an object of class tssdr.

### Usage

```
## S3 method for class 'tssdr'
components(x, ...)
```

### Arguments

x                      Object of class tssdr.  
 ...                    Further arguments to be passed to or from methods.

### Author(s)

Markus Matilainen

### See Also

[tssdr](#)

---

FixNA                      *The FixNA method for Blind Source Separation*

---

### Description

The FixNA (Shi et al., 2009) and FixNA2 (Matilainen et al., 2017) methods for blind source separation problem. It is used for time series with stochastic volatility. These methods are alternatives to vSOBI method.

### Usage

```
FixNA(X, ...)

## Default S3 method:
FixNA(X, k = 1:12, eps = 1e-06, maxiter = 1000, G = "pow", method = "FixNA",
      ordered = FALSE, acfk = NULL, original = TRUE, ...)
## S3 method for class 'ts'
FixNA(X, ...)
```

**Arguments**

X	A numeric matrix or a multivariate time series object of class <code>ts</code> . Missing values are not allowed.
k	A vector of lags. It can be any non-zero positive integer, or a vector consisting of them. Default is <code>1:12</code> .
eps	Convergence tolerance.
maxiter	The maximum number of iterations.
G	Function $G(x)$ . The choices are <code>pow</code> (default) and <code>lcosh</code> .
method	The method to be used. The choices are <code>FixNA</code> (default) and <code>FixNA2</code> .
ordered	Whether to order components according to their volatility. Default is <code>FALSE</code> .
acfk	A vector of lags to be used in testing the presence of serial autocorrelation. Applicable only if <code>ordered = TRUE</code> .
original	Whether to return the original components or their residuals based on ARMA fit. Default is <code>TRUE</code> , i.e. the original components are returned. Applicable only if <code>ordered = TRUE</code> .
...	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 algorithm for method `FixNA` finds an orthogonal matrix  $\mathbf{U}$  by maximizing

$$\mathbf{D}_1(\mathbf{U}) = \sum_{k=1}^K \mathbf{D}_{1k}(\mathbf{U}) = \sum_{k=1}^K \sum_{i=1}^p \frac{1}{T-k} \sum_{t=1}^{T-k} [G(\mathbf{u}'_i \mathbf{Y}_t) G(\mathbf{u}'_i \mathbf{Y}_{t+k})]$$

and the algorithm for method `FixNA2` finds an orthogonal matrix  $\mathbf{U}$  by maximizing

$$\begin{aligned} \mathbf{D}_2(\mathbf{U}) &= \sum_{k=1}^K \mathbf{D}_{2k}(\mathbf{U}) \\ &= \sum_{k=1}^K \sum_{i=1}^p \left| \frac{1}{T-k} \sum_{t=1}^{T-k} [G(\mathbf{u}'_i \mathbf{Y}_t) G(\mathbf{u}'_i \mathbf{Y}_{t+k})] - \left( \frac{1}{T-k} \right)^2 \sum_{t=1}^{T-k} [G(\mathbf{u}'_i \mathbf{Y}_t)] \sum_{t=1}^{T-k} [G(\mathbf{u}'_i \mathbf{Y}_{t+k})] \right|. \end{aligned}$$

For function  $G(x)$  the choices are  $x^2$  and  $\log(\cosh(x))$ .

The algorithm works iteratively starting with  $\text{diag}(p)$  as an initial value for an orthogonal matrix  $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p)'$ .

Matrix  $\mathbf{T}_{mik}$  is a partial derivative of  $\mathbf{D}_{mk}(\mathbf{U})$ , for  $m = 1, 2$ , with respect to  $\mathbf{u}_i$ . Then  $\mathbf{T}_{mk} = (\mathbf{T}_{m1k}, \dots, \mathbf{T}_{mpk})'$ , where  $p$  is the number of columns in  $\mathbf{Y}$ , and  $\mathbf{T}_m = \sum_{k=1}^K \mathbf{T}_{mk}$ . The update for the orthogonal matrix  $\mathbf{U}_{new} = (\mathbf{T}_m \mathbf{T}'_m)^{-1/2} \mathbf{T}_m$  is calculated at each iteration step. The algorithm stops when

$$\|\mathbf{U}_{new} - \mathbf{U}_{old}\|$$

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

For `ordered = TRUE` the function orders the sources according to their volatility. First a possible linear autocorrelation is removed using `auto.arima`. Then a squared autocorrelation test is performed for the sources (or for their residuals, when linear correlation is present). The sources are then put in a decreasing order according to the value of the test statistic of the squared autocorrelation test. For more information, see `lbtest`.

### Value

A list with class 'bssvol' (inherits from class 'bss') containing the following components:

<code>W</code>	The estimated unmixing matrix.
<code>k</code>	The vector of the used lags.
<code>S</code>	The estimated sources as time series object standardized to have mean 0 and unit variances. If <code>ordered = TRUE</code> , then components are ordered according to their volatility.

If `ordered = TRUE`, then also the following components included in the list:

<code>fits</code>	The ARMA fits for the components with linear autocorrelation.
<code>armaeff</code>	A logical vector. Has value 1 if ARMA fit was done to the corresponding component.
<code>linTS</code>	The value of the modified Ljung-Box test statistic for each component.
<code>linP</code>	P-value based on the modified Ljung-Box test statistic for each component.
<code>volTS</code>	The value of the volatility clustering test statistic.
<code>volP</code>	P-value based on the volatility clustering test statistic.

### Author(s)

Markus Matilainen

### References

- Hyv?rinen, A. (2001), *Blind Source Separation by Nonstationarity of Variance: A Cumulant-Based Approach*, IEEE Transactions on Neural Networks, 12(6): 1471–1474.
- Matilainen, M., Miettinen, J., Nordhausen, K., Oja, H. and Taskinen, S. (2017), *On Independent Component Analysis with Stochastic Volatility Models*, Austrian Journal of Statistics, 46(3–4), 57–66.
- Shi, Z., Jiang, Z. and Zhou, F. (2009), *Blind Source Separation with Nonlinear Autocorrelation and Non-Gaussianity*, Journal of Computational and Applied Mathematics, 223(1): 908–915.

### See Also

`vSOBI`, `lbtest`, `auto.arima`

**Examples**

```

library(stochvol)
n <- 10000
A <- matrix(rnorm(9), 3, 3)

# simulate SV models
s1 <- svsim(n, mu = -10, phi = 0.8, sigma = 0.1)$y
s2 <- svsim(n, mu = -10, phi = 0.9, sigma = 0.2)$y
s3 <- svsim(n, mu = -10, phi = 0.95, sigma = 0.4)$y

# create a daily time series
X <- ts(cbind(s1, s2, s3) %*% t(A), end = c(2015, 338), frequency = 365.25)

res <- FixNA(X)
res
coef(res)
plot(res)
head(bss.components(res))

MD(res$W, A) # Minimum Distance Index, should be close to zero

```

gFOBI

*Generalized FOBI***Description**

The gFOBI method for blind source separation problem. It is designed for time series with stochastic volatility. The method is a generalization of FOBI, which is a method designed for iid data.

**Usage**

```

gFOBI(X, ...)

## Default S3 method:
gFOBI(X, k = 0:12, eps = 1e-06, maxiter = 100, method = "frjd",
      na.action = na.fail, weight = NULL,
      ordered = FALSE, acfk = NULL, original = TRUE, ...)

## S3 method for class 'ts'
gFOBI(X, ...)

```

**Arguments**

**X** A numeric matrix or a multivariate time series object of class `ts`. Missing values are not allowed.

**k** A vector of lags. It can be any non-negative integer, or a vector consisting of them. Default is `0:12`. If  $k = 0$ , this method reduces to [FOBI](#).



eps	Convergence tolerance.
maxiter	The maximum number of iterations.
method	The method to use for the joint diagonalization. The options are <a href="#">rjd</a> and <a href="#">frjd</a> . Default is <a href="#">frjd</a> .
na.action	A function which indicates what should happen when the data contain 'NA's. Default is to fail.
weight	A vector of length k to give weight to the different matrices in joint diagonalization. If NULL, all matrices have equal weight.
ordered	Whether to order components according to their volatility. Default is FALSE.
acfk	A vector of lags to be used in testing the presence of serial autocorrelation. Applicable only if ordered = TRUE.
original	Whether to return the original components or their residuals based on ARMA fit. Default is TRUE, i.e. the original components are returned. Applicable only if ordered = TRUE.
...	Other arguments passed on to <code>auto.arima</code> function.

### 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  $\mathbf{X}$ . Algorithm first calculates

$$\widehat{\mathbf{B}}_k^{ij}(\mathbf{Y}) = \frac{1}{T-k} \sum_{t=1}^T [\mathbf{Y}_{t+k} \mathbf{Y}_t' \mathbf{E}^{ij} \mathbf{Y}_t \mathbf{Y}_{t+k}']$$

and then

$$\widehat{\mathbf{B}}_k(\mathbf{Y}) = \sum_{i=1}^p \widehat{\mathbf{B}}_k^{ii}(\mathbf{Y}).$$

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

$$\sum_{k=0}^K \|\text{diag}(\mathbf{U} \widehat{\mathbf{B}}_k(\mathbf{Y}) \mathbf{U}')\|^2.$$

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

For `ordered = TRUE` the function orders the sources according to their volatility. First a possible linear autocorrelation is removed using `auto.arima`. Then a squared autocorrelation test is performed for the sources (or for their residuals, when linear correlation is present). The sources are then put in a decreasing order according to the value of the test statistic of the squared autocorrelation test. For more information, see [lbttest](#).

### Value

A list with class 'bssvol' (inherits from class 'bss') containing the following components:

w	The estimated unmixing matrix.
k	The vector of the used lags.

**S** The estimated sources as time series object standardized to have mean 0 and unit variances. If `ordered = TRUE`, then components are ordered according to their volatility.

If `ordered = TRUE`, then also the following components included in the list:

**fits** The ARMA fits for the components with linear autocorrelation.  
**armaeff** A logical vector. Has value 1 if ARMA fit was done to the corresponding component.  
**linTS** The value of the modified Ljung-Box test statistic for each component.  
**linP** P-value based on the modified Ljung-Box test statistic for each component.  
**volTS** The value of the volatility clustering test statistic.  
**volP** P-value based on the volatility clustering test statistic.

### Author(s)

Markus Matilainen, Klaus Nordhausen

### References

Cardoso, J.-F., (1989), *Source Separation Using Higher Order Moments*, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2109–2112.

Matilainen, M., Nordhausen, K. and Oja, H. (2015), *New Independent Component Analysis Tools for Time Series*, *Statistics & Probability Letters*, 105, 80–87.

### See Also

[FOBI](#), [frjd](#), [lbttest](#), [auto.arima](#)

### Examples

```
library(stochvol)
n <- 10000
A <- matrix(rnorm(9), 3, 3)

# simulate SV models
s1 <- svsim(n, mu = -10, phi = 0.8, sigma = 0.1)$y
s2 <- svsim(n, mu = -10, phi = 0.9, sigma = 0.2)$y
s3 <- svsim(n, mu = -10, phi = 0.95, sigma = 0.4)$y

X <- cbind(s1, s2, s3) %*% t(A)

res <- gFOBI(X)
res
coef(res)
plot(res)
head(bss.components(res))

MD(res$W, A) # Minimum Distance Index, should be close to zero
```

gJADE

*Generalized JADE***Description**

The gJADE method for blind source separation problem. It is designed for time series with stochastic volatility. The method is a generalization of JADE, which is a method for blind source separation problem using only marginal information.

**Usage**

```
gJADE(X, ...)

## Default S3 method:
gJADE(X, k = 0:12, eps = 1e-06, maxiter = 100, method = "frjd",
      na.action = na.fail, weight = NULL,
      ordered = FALSE, acfk = NULL, original = TRUE, ...)
## S3 method for class 'ts'
gJADE(X, ...)
```

**Arguments**

X	A numeric matrix or a multivariate time series object of class <code>ts</code> . Missing values are not allowed.
k	A vector of lags. It can be any non-negative integer, or a vector consisting of them. Default is <code>0:12</code> . If $k = 0$ , this method reduces to <code>JADE</code> .
eps	Convergence tolerance.
maxiter	The maximum number of iterations.
method	The method to use for the joint diagonalization. The options are <code>rjd</code> and <code>frjd</code> . Default is <code>frjd</code> .
na.action	A function which indicates what should happen when the data contain 'NA's. Default is to fail.
weight	A vector of length k to give weight to the different matrices in joint diagonalization. If <code>NULL</code> , all matrices have equal weight.
ordered	Whether to order components according to their volatility. Default is <code>FALSE</code> .
acfk	A vector of lags to be used in testing the presence of serial autocorrelation. Applicable only if <code>ordered = TRUE</code> .
original	Whether to return the original components or their residuals based on ARMA fit. Default is <code>TRUE</code> , i.e. the original components are returned. Applicable only if <code>ordered = TRUE</code> .
...	Other arguments passed on to <code>auto.arima</code> function.

### 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 matrix  $\widehat{\mathbf{C}}_k^{ij}(\mathbf{Y})$  is of the form

$$\widehat{\mathbf{C}}_k^{ij}(\mathbf{Y}) = \widehat{\mathbf{B}}_k^{ij}(\mathbf{Y}) - \mathbf{S}_k(\mathbf{Y})(\mathbf{E}^{ij} + \mathbf{E}^{ji})\mathbf{S}_k(\mathbf{Y})' - \text{trace}(\mathbf{E}^{ij})\mathbf{I}_p,$$

for  $i, j = 1, \dots, p$ , where  $\mathbf{S}_k(\mathbf{Y})$  is the lagged sample covariance matrix of  $\mathbf{Y}$  for lag  $k = 1, \dots, K$ ,  $\mathbf{E}^{ij}$  is a matrix where element  $(i, j)$  equals to 1 and all other elements are 0,  $\mathbf{I}_p$  is an identity matrix of order  $p$  and  $\widehat{\mathbf{B}}_k^{ij}(\mathbf{Y})$  is as in [gFOBI](#).

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

$$\sum_{i=1}^p \sum_{j=1}^p \sum_{k=0}^K \|\text{diag}(\mathbf{U}\widehat{\mathbf{C}}_k^{ij}(\mathbf{Y})\mathbf{U}')\|^2.$$

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

For `ordered = TRUE` the function orders the sources according to their volatility. First a possible linear autocorrelation is removed using [auto.arima](#). Then a squared autocorrelation test is performed for the sources (or for their residuals, when linear correlation is present). The sources are then put in a decreasing order according to the value of the test statistic of the squared autocorrelation test. For more information, see [lbtest](#).

### Value

A list with class 'bssvol' (inherits from class 'bss') containing the following components:

<code>W</code>	The estimated unmixing matrix.
<code>k</code>	The vector of the used lags.
<code>S</code>	The estimated sources as time series object standardized to have mean 0 and unit variances. If <code>ordered = TRUE</code> , then components are ordered according to their volatility.

If `ordered = TRUE`, then also the following components included in the list:

<code>fits</code>	The ARMA fits for the components with linear autocorrelation.
<code>armaeff</code>	A logical vector. Has value 1 if ARMA fit was done to the corresponding component.
<code>linTS</code>	The value of the modified Ljung-Box test statistic for each component.
<code>linP</code>	P-value based on the modified Ljung-Box test statistic for each component.
<code>volTS</code>	The value of the volatility clustering test statistic.
<code>volP</code>	P-value based on the volatility clustering test statistic.

### Author(s)

Klaus Nordhausen, Markus Matilainen

## References

- Cardoso, J.-F., Souloumiac, A., (1993). *Blind Beamforming for Non-Gaussian Signals*, in: IEE-Proceedings-F, volume 140, pp. 362–370.
- Matilainen, M., Nordhausen, K. and Oja, H. (2015), *New Independent Component Analysis Tools for Time Series*, *Statistics & Probability Letters*, 105, 80–87.

## See Also

[frjd](#), [JADE](#), [gFOBI](#), [lbtest](#), [auto.arima](#)

## Examples

```
library(stochvol)
n <- 10000
A <- matrix(rnorm(9), 3, 3)

# simulate SV models
s1 <- svsim(n, mu = -10, phi = 0.8, sigma = 0.1)$y
s2 <- svsim(n, mu = -10, phi = 0.9, sigma = 0.2)$y
s3 <- svsim(n, mu = -10, phi = 0.95, sigma = 0.4)$y

X <- cbind(s1, s2, s3) %*% t(A)

res <- gJADE(X)
res
coef(res)
plot(res)
head(bss.components(res))

MD(res$W, A) # Minimum Distance Index, should be close to zero
```

---

gSOBI

*Generalized SOBI*

---

## Description

The gSOBI method for the blind source separation problem. It is designed for multivariate time series, where its components can be with or without stochastic volatility. The method is a combination of SOBI and vSOBI with  $G(x) = x^2$  as a nonlinearity function.

## Usage

```
gSOBI(X, k1 = 1:12, k2 = 1:3, b = 0.9, eps = 1e-06, maxiter = 1000, ordered = FALSE,
      acfk = NULL, original = TRUE, ...)
```

**Arguments**

X	A numeric matrix or a multivariate time series object of class <code>ts</code> . Missing values are not allowed.
k1	A vector of lags for SOBI part. It can be any non-zero positive integer, or a vector consisting of them. Default is 1:12.
k2	A vector of lags for vSOBI part. It can be any non-zero positive integer, or a vector consisting of them. Default is 1:3.
b	The weight for the SOBI part, $1 - b$ for the vSOBI part. Default is 0.9.
eps	Convergence tolerance.
maxiter	The maximum number of iterations.
ordered	Whether to order components according to their volatility. Default is FALSE.
acfk	A vector of lags to be used in testing the presence of serial autocorrelation. Applicable only if ordered = TRUE.
original	Whether to return the original components or their residuals based on ARMA fit. Default is TRUE, i.e. the original components are returned. Applicable only if ordered = TRUE.
...	Other arguments passed on to <code>auto.arima</code> function.

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

$$\mathbf{D}(\mathbf{U}) = b \sum_{k_1=1}^{K_1} \mathbf{D}_{k_1}(\mathbf{U}) + (1 - b) \sum_{k_2=1}^{K_2} \mathbf{D}_{k_2}(\mathbf{U}),$$

where SOBI part

$$\mathbf{D}_{k_1} = \sum_{i=1}^p \left( \frac{1}{T - k_1} \sum_{t=1}^{T-k_1} [(\mathbf{u}'_i \mathbf{Y}_t)(\mathbf{u}'_i \mathbf{Y}_{t+k_1})] \right)^2.$$

and vSOBI part

$$\mathbf{D}_{k_2} = \sum_{i=1}^p \left( \frac{1}{T - k_2} \sum_{t=1}^{T-k_2} [(\mathbf{u}'_i \mathbf{Y}_t)^2 (\mathbf{u}'_i \mathbf{Y}_{t+k_2})^2] - \left( \frac{1}{T - k_2} \right)^2 \sum_{t=1}^{T-k_2} [(\mathbf{u}'_i \mathbf{Y}_t)^2] \sum_{t=1}^{T-k_2} [(\mathbf{u}'_i \mathbf{Y}_{t+k_2})^2] \right)^2$$

where  $b \in [0, 1]$ .

The algorithm works iteratively starting with  $\text{diag}(p)$  as an initial value for an orthogonal matrix  $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p)'$ .

Matrix  $\mathbf{T}_{ikj}$  is a partial derivative of  $\mathbf{D}_{kj}(\mathbf{U})$ , where  $j = 1, 2$ , with respect to  $\mathbf{u}_i$ . Then  $\mathbf{T}_{kj} = (\mathbf{T}_{1kj}, \dots, \mathbf{T}_{pkj})'$ , where  $p$  is the number of columns in  $\mathbf{Y}$ , and  $\mathbf{T}_j = \sum_{k_j=1}^{K_j} \mathbf{T}_{kj}$ , for  $j = 1, 2$ . Finally  $\mathbf{T} = b\mathbf{T}_1 + (1 - b)\mathbf{T}_2$ .

The update for the orthogonal matrix  $\mathbf{U}_{new} = (\mathbf{T}\mathbf{T}')^{-1/2}\mathbf{T}$  is calculated at each iteration step. The algorithm stops when

$$\|\mathbf{U}_{new} - \mathbf{U}_{old}\|$$

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

For `ordered = TRUE` the function orders the sources according to their volatility. First a possible linear autocorrelation is removed using `auto.arima`. Then a squared autocorrelation test is performed for the sources (or for their residuals, when linear correlation is present). The sources are then put in a decreasing order according to the value of the test statistic of the squared autocorrelation test. For more information, see `lbtest`.

### Value

A list with class 'bssvol' (inherits from class 'bss') containing the following components:

<code>W</code>	The estimated unmixing matrix.
<code>k1</code>	The vector of the used lags for the SOBI part.
<code>k2</code>	The vector of the used lags for the vSOBI part.
<code>S</code>	The estimated sources as time series object standardized to have mean 0 and unit variances. If <code>ordered = TRUE</code> , then components are ordered according to their volatility.

If `ordered = TRUE`, then also the following components included in the list:

<code>fits</code>	The ARMA fits for the components with linear autocorrelation.
<code>armaeff</code>	A logical vector. Has value 1 if ARMA fit was done to the corresponding component.
<code>linTS</code>	The value of the modified Ljung-Box test statistic for each component.
<code>linP</code>	P-value based on the modified Ljung-Box test statistic for each component.
<code>volTS</code>	The value of the volatility clustering test statistic.
<code>volP</code>	P-value based on the volatility clustering test statistic.

### Author(s)

Markus Matilainen, Jari Miettinen

### References

- Belouchrani, A., Abed-Meriam, K., Cardoso, J.F. and Moulines, R. (1997), *A blind source separation technique using second-order statistics*, IEEE Transactions on Signal Processing, 434–444.
- Matilainen, M., Miettinen, J., Nordhausen, K., Oja, H. and Taskinen, S. (2017), *On Independent Component Analysis with Stochastic Volatility Models*, Austrian Journal of Statistics, 46(3–4), 57–66.
- Miettinen, M., Matilainen, M., Nordhausen, K. and Taskinen, S. (2017), *Extracting Conditionally Heteroscedastic Components Using ICA*, Submitted.

**See Also**

[SOBI](#), [vSOBI](#), [lbtest](#), [auto.arima](#)

**Examples**

```
library(stochvol)
n <- 10000
A <- matrix(rnorm(9), 3, 3)

# simulate SV models
s1 <- svsim(n, mu = -10, phi = 0.8, sigma = 0.1)$y
s2 <- svsim(n, mu = -10, phi = 0.9, sigma = 0.2)$y
s3 <- svsim(n, mu = -10, phi = 0.95, sigma = 0.4)$y

# create a daily time series
X <- ts(cbind(s1, s2, s3) %*% t(A), end = c(2015, 338), frequency = 365.25)

res <- gSOBI(X, 1:4, 1:2, 0.99)
res$W
coef(res)
plot(res)
head(bss.components(res))

MD(res$W, A) # Minimum Distance Index, should be close to zero
```

---

 lbtest

*Modified Ljung-Box test and volatility clustering test for time series.*

---

**Description**

Modified Ljung-Box test and volatility clustering test time series. The modified Ljung-Box test checks whether there is linear autocorrelation in the time series. The volatility clustering test checks whether the time series has squared autocorrelation, which would indicate a presence of volatility clustering.

**Usage**

```
lbtest(X, k, type = "squared")
```

**Arguments**

X	A numeric matrix or a multivariate time series object of class <code>ts</code> . Missing values are not allowed.
k	A vector of lags.
type	The type of the autocorrelation test. Options are Modified Ljung-Box test (linear) or volatility clustering test (squared) autocorrelation. Default is squared.



### Details

Assume all the individual time series  $X_i$  in  $\mathbf{X}$  with  $T$  observations are scaled to have variance 1.

Then the modified Ljung-Box test statistic for testing the existence of linear autocorrelation in  $X_i$  (option = "linear") is

$$T \sum_{j \in k} \left( \sum_{t=1}^T (X_{it} X_{i,t+j}) / (T-j) \right)^2 / V_j.$$

Here

$$V_j = \sum_{t=1}^{n-j} \frac{x_t^2 x_{t+j}^2}{n-j} + 2 \sum_{k=1}^{n-j-1} \frac{n-k}{n} \sum_{s=1}^{n-k-j} \frac{x_s x_{s+j} x_{s+k} x_{s+k+j}}{n-k-j}.$$

The volatility clustering test statistic (option = "squared") is

$$T \sum_{j \in k} \left( \sum_{t=1}^T (X_{it}^2 X_{i,t+j}^2) / (T-j) - 1 \right)^2$$

Test statistic related to each time series  $X_i$  is then compared to  $\chi^2$ -distribution with length(k) degrees of freedom, and the corresponding p-values are produced. Small p-value indicates the existence of autocorrelation.

### Value

A list with class 'lbtest' containing the following components:

TS	The values of the test statistic for each component of X as a vector.
p_val	The p-values based on the test statistic for each component of X as a vector.
Xname	The name of the data used as a character string.
varnames	The names of the variables used as a character string vector.
k	The lags used for testing the serial autocorrelation as a vector.
K	The total number of lags used for testing the serial autocorrelation.
type	The type of the autocorrelation test.

### Author(s)

Markus Matilainen, Jari Miettinen

### References

Miettinen, M., Matilainen, M., Nordhausen, K. and Taskinen, S. (2017), ???, ???.

### See Also

[print.lbtest](#), [FixNA](#), [gFOBI](#), [gJADE](#), [vSOBI](#), [gSOBI](#)

## Examples

```
library(stochvol)
n <- 10000
s1 <- svsim(n, mu = -10, phi = 0.95, sigma = 0.1)$y
s2 <- rnorm(n)
S <- cbind(s1, s2)

lbttest(S, 1:3, type = "squared")
# First p-value should be very close to zero, as there exists stochastic volatility
```

---

plot.summary.tssdr      *Plotting an Object of class summary.tssdr*

---

## Description

Plots the response and the estimated directions (sources) resulting from a summary of a tssdr method. The directions are the chosen directions from the chosen tssdr method. The multivariate time series is a time series object, which is passed on to `plot.ts`.

## Usage

```
## S3 method for class 'summary.tssdr'
plot(x, main = "The response and the chosen directions", ...)
```

## Arguments

<code>x</code>	An object of class <code>summary.tssdr</code> .
<code>main</code>	A title for the time series plot.
<code>...</code>	Further arguments to be passed to or from methods.

## Author(s)

Markus Matilainen

## See Also

[plot.ts](#), [summary.tssdr](#)

---

plot.tssdr	<i>Plotting an object of class tssdr</i>
------------	--

---

**Description**

Plots the response and the estimated directions (sources) resulting from a tssdr method. The multi-variate time series is a time series object, which is passed on to plot.ts.

**Usage**

```
## S3 method for class 'tssdr'  
plot(x, main = "The response and the directions", ...)
```

**Arguments**

x	An object of class tssdr.
main	A title for the time series plot.
...	Further arguments to be passed to or from methods.

**Author(s)**

Markus Matilainen

**See Also**

[plot.ts](#), [tssdr](#)

---

print.bssvol	<i>Printing an object of class bssvol</i>
--------------	---

---

**Description**

Prints an object of class bssvol. It prints the unmixing matrix  $W$  and the used lags  $k$ .

**Usage**

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

**Arguments**

x	An object of class bssvol.
...	Further arguments to be passed to or from methods.

**Author(s)**

Markus Matilainen

**See Also**

[print.bss](#)

---

print.lbtest	<i>Printing an object of class lbtest</i>
--------------	---

---

**Description**

Prints an object of class lbtest.

**Usage**

```
## S3 method for class 'lbtest'  
print(x, digits = 3, ...)
```

**Arguments**

x	An object of class lbtest.
digits	The chosen number of digits.
...	Further arguments to be passed to or from methods.

**Author(s)**

Markus Matilainen

**See Also**

[lbtest](#)

---

print.summary.tssdr	<i>Printing an object of class summary.tssdr</i>
---------------------	--

---

**Description**

Prints an object of class summary.tssdr. It prints all the elements of the list of class summary.tssdr except the component S, which is the matrix of the chosen directions or a vector, if only one direction is chosen.

**Usage**

```
## S3 method for class 'summary.tssdr'  
print(x, digits = 3, ...)
```

**Arguments**

- `x` An object of class `summary.tssdr`.
- `digits` The chosen number of digits.
- `...` Further arguments to be passed to or from methods.

**Author(s)**

Markus Matilainen

**See Also**

[summary.tssdr](#)

---

`print.tssdr` *Printing an object of class tssdr*

---

**Description**

Prints an object of class `tssdr`. It prints the signal separation matrix `W`, matrix `L` to decide which lags and directions are important, and the vector `k`, i.e. the used lag(s).

**Usage**

```
## S3 method for class 'tssdr'  
print(x, digits = 3, ...)
```

**Arguments**

- `x` An object of class `tssdr`.
- `digits` The chosen number of digits.
- `...` Further arguments to be passed to or from methods.

**Author(s)**

Markus Matilainen

**See Also**

[tssdr](#)

PVC

*A modified algorithm for Principal Volatility Component estimator***Description**

PVC (Principal Volatility Component) estimator for the blind source separation problem. This method is a modified version of PVC by Hu and Tsay (2014).

**Usage**

```
PVC(X, k = 1:12, ordered = FALSE, acfk = NULL, original = TRUE, ...)
```

**Arguments**

X	A numeric matrix or a multivariate time series object of class <code>ts</code> . Missing values are not allowed.
k	A vector of lags. It can be any non-zero positive integer, or a vector consisting of them. Default is 1:12.
ordered	Whether to order components according to their volatility. Default is FALSE.
acfk	A vector of lags to be used in testing the presence of serial autocorrelation. Applicable only if ordered = TRUE.
original	Whether to return the original components or their residuals based on ARMA fit. Default is TRUE, i.e. the original components are returned. Applicable only if ordered = TRUE.
...	Other arguments passed on to <code>auto.arima</code> function.

**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  $\mathbf{X}$ . Then for each lag  $k$  we calculate

$$\widehat{Cov}(\mathbf{Y}_t \mathbf{Y}_t', Y_{ij,t-k}) = \frac{1}{T} \sum_{t=k+1}^T \left( \mathbf{Y}_t \mathbf{Y}_t' - \frac{1}{T-k} \sum_{t=k+1}^T \mathbf{Y}_t \mathbf{Y}_t' \right) \left( Y_{ij,t-k} - \frac{1}{T-k} \sum_{t=k+1}^T Y_{ij,t-k} \right),$$

where  $Y_{ij,t-k} = Y_{i,t-k} Y_{j,t-k}$ ,  $i, j = 1, \dots, p$ . Then

$$\mathbf{g}_k(\mathbf{Y}) = \sum_{i=1}^p \sum_{j=1}^p (\widehat{Cov}(\mathbf{Y}_t \mathbf{Y}_t', Y_{ij,t-k}))^2.$$

Thus the generalized kurtosis matrix is

$$\mathbf{G}_K(\mathbf{Y}) = \sum_{k=1}^K \mathbf{g}_k(\mathbf{Y}),$$

where  $k = 1, \dots, K$  is the set of chosen lags. Then  $\mathbf{U}$  is the matrix with eigenvectors of  $\mathbf{G}_K(\mathbf{Y})$  as its rows. The final unmixing matrix is then  $\mathbf{W} = \mathbf{U}\mathbf{S}^{-1/2}$ , where the average value of each row is set to be positive.

For `ordered = TRUE` the function orders the sources according to their volatility. First a possible linear autocorrelation is removed using `auto.arima`. Then a squared autocorrelation test is performed for the sources (or for their residuals, when linear correlation is present). The sources are then put in a decreasing order according to the value of the test statistic of the squared autocorrelation test. For more information, see `lbtest`.

### Value

A list with class 'bssvol' (inherits from class 'bss') containing the following components:

<code>W</code>	The estimated unmixing matrix.
<code>k</code>	The vector of the used lags.
<code>S</code>	The estimated sources as time series object standardized to have mean 0 and unit variances. If <code>ordered = TRUE</code> , then components are ordered according to their volatility.

If `ordered = TRUE`, then also the following components included in the list:

<code>fits</code>	The ARMA fits for the components with linear autocorrelation.
<code>armaeff</code>	A logical vector. Has value 1 if ARMA fit was done to the corresponding component.
<code>linTS</code>	The value of the modified Ljung-Box test statistic for each component.
<code>linP</code>	P-value based on the modified Ljung-Box test statistic for each component.
<code>volTS</code>	The value of the volatility clustering test statistic.
<code>volP</code>	P-value based on the volatility clustering test statistic.

### Author(s)

Jari Miettinen, Markus Matilainen

### References

- Miettinen, M., Matilainen, M., Nordhausen, K. and Taskinen, S. (2017), *Extracting Conditionally Heteroscedastic Components Using ICA*, Submitted.
- Hu, Y.-P. & Tsay, R. S. (2014), *Principal Volatility Component Analysis*, Journal of Business & Economic Statistics, 32(2), 153–164.

### See Also

[comVol](#), [gSOBI](#), [lbtest](#), [auto.arima](#)

**Examples**

```

library(stochvol)
n <- 10000
A <- matrix(rnorm(9), 3, 3)

# simulate SV models
s1 <- svsim(n, mu = -10, phi = 0.8, sigma = 0.1)$y
s2 <- svsim(n, mu = -10, phi = 0.9, sigma = 0.2)$y
s3 <- svsim(n, mu = -10, phi = 0.95, sigma = 0.4)$y

# create a daily time series
X <- ts(cbind(s1, s2, s3) %*% t(A), end = c(2015, 338), frequency = 365.25)

res <- PVC(X)
res
coef(res)
plot(res)
head(bss.components(res))

MD(res$W, A) # Minimum Distance Index, should be close to zero

```

---

summary.tssdr

*Summary of an object of class tssdr*


---

**Description**

Gives a summary of an object of class tssdr. It includes different types of methods to select the number of directions (sources) and lags.

**Usage**

```

## S3 method for class 'tssdr'
summary(object, type = "rectangle", thres = 0.8, ...)

```

**Arguments**

object	An object of class tssdr.
type	Method for choosing the important lags and directions. The choices are rectangle, alllag, alldir and big. Default is rectangle.
thres	The threshold value for choosing the lags and directions. Default is 0.8.
...	Further arguments to be passed to or from methods.



## Details

The sum of values of  $k_0 \times p_0$  matrix  $\mathbf{L}$  of object is 1. The values of the matrix are summed together in ways detailed below, until the value is at least  $\pi$  (thres). Let  $\lambda_{ij}$  be the element  $(i, j)$  of the matrix  $\mathbf{L}$ .

For alllag:  $k = k_0$  and  $p$  is the smallest value for which  $\sum_{i=1}^p \lambda_{ij} \geq \pi$ .

For alldir:  $p = p_0$  and  $k$  is the smallest value for which  $\sum_{j=1}^k \lambda_{ij} \geq \pi$

For rectangle:  $k$  and  $p$  are values such that their product  $kp$  is the smallest for which  $\sum_{i=1}^p \sum_{j=1}^k \lambda_{ij} \geq \pi$

For big:  $r$  is the smallest value of elements  $(i_1, j_1), \dots, (i_r, j_r)$  for which  $\sum_{k=1}^r \lambda_{i_k, j_k} \geq \pi$

Note that when printing a summary.tssdr object, all elements except the component  $\mathbf{S}$ , which is the matrix of the chosen directions or a vector if there is only one direction, are printed.

## Value

A list with class 'summary.tssdr' containing the following components:

W	The estimated signal separation matrix
L	The Lambda matrix for choosing lags and directions.
S	The estimated directions as time series object standardized to have mean 0 and unit variances.
type	The method for choosing the important lags and directions.
algorithm	The used algorithm as a character string.
yname	The name for the response time series $y$ .
Xname	The name for the predictor time series $\mathbf{X}$ .
k	The chosen number of lags (not for type = "big").
p	The chosen number of directions (not for type = "big").
pk	The chosen lag-direction combinations (for type = "big" only).

## Author(s)

Markus Matilainen

## References

Matilainen M., Croux C., Nordhausen K. & Oja H. (2017), *Supervised Dimension Reduction for Multivariate Time Series*, *Econometrics and Statistics*, 4, 57–69.

## See Also

[tssdr](#)

**Examples**

```

n <- 10000
A <- matrix(rnorm(9), 3, 3)

x1 <- arima.sim(n = n, list(ar = 0.2))
x2 <- arima.sim(n = n, list(ar = 0.8))
x3 <- arima.sim(n = n, list(ar = 0.3, ma = -0.4))
eps2 <- rnorm(n - 1)
y <- 2*x1[1:(n - 1)] + 3*x2[1:(n - 1)] + eps2
X <- ((cbind(x1, x2, x3))[2:n, ]) %>% t(A)

res2 <- tssdr(y, X, algorithm = "TSIR")
res2
summ2 <- summary(res2, thres = 0.5)
summ2
summary(res2) #Chooses more lags with larger threshold
summary(res2, type = "alllag") #Chooses all lags
summary(res2, type = "alldir", thres = 0.5) #Chooses all directions
summary(res2, type = "big", thres = 0.5) #Same choices than in summ2

```

tssdr

*Supervised dimension reduction for multivariate time series***Description**

Supervised dimension reduction for multivariate time series data. There are three different algorithms to choose from. TSIR is a time series version of Sliced Inverse Regression (SIR), TSAVE is a time series version of Sliced Average Variance Estimate (TSAVE) and TSSH is a hybrid of TSIR and TSAVE.

**Usage**

```

tssdr(y, X, algorithm = "TSIR", k = 1:12, H = 10, weight = 0.5, method = "frjd",
      eps = 1e-06, maxiter = 1000, ...)

```

**Arguments**

y	A numeric vector or time series object of class <code>ts</code> . Missing values are not allowed.
X	A numeric matrix or a multivariate time series object of class <code>ts</code> . Missing values are not allowed.
algorithm	Algorithm to be used. The options are TSIR, TSAVE and TSSH. Default is TSIR.
k	A vector of lags. It can be any non-zero positive integer, or a vector consisting of them. Default is 1:12.
H	The number of slices. If TSSH is used, $H$ is a 2-vector; the first element is used for TSIR part and the second for TSAVE part. Default is $H = 10$ .

weight	Weight $0 \leq a \leq 1$ for the hybrid method TSSH only. With $a = 1$ it reduces to TSAVE and with $a = 0$ to TSIR. Default is $a = 0.5$ .
method	The method to use for the joint diagonalization. The options are <code>rjd</code> and <code>frjd</code> . Default is <code>frjd</code> .
eps	Convergence tolerance.
maxiter	The maximum number of iterations.
...	Other arguments passed on to the chosen joint diagonalization method.

### Details

Assume that the  $p$ -variate time series  $\mathbf{Z}$  with  $T$  observations is whitened, i.e.  $\mathbf{Z} = \mathbf{S}^{-1/2}(\mathbf{X}_t - \frac{1}{T} \sum_{t=1}^T \mathbf{X}_t)$ , where  $\mathbf{S}$  is a sample covariance matrix of  $\mathbf{X}$ . Divide  $y$  into  $H$  disjoint intervals (slices) by its empirical quantiles.

For each lag  $j$ , denote  $y_j$  for a vector of the last  $n - j$  values of the sliced  $y$ . Also denote  $\mathbf{Z}_j$  for the first  $n - j$  observations of  $\mathbf{Z}$ . Then  $\mathbf{Z}_{jh}$  are the disjoint slices of  $\mathbf{Z}_j$  according to the values of  $y_j$ .

Let  $T_{jh}$  be the number of observations in  $\mathbf{Z}_{jh}$ . Write  $\hat{\mathbf{A}}_{jh} = \frac{1}{T_{jh}} \sum_{t=1}^{T_{jh}} (\mathbf{Z}_{jh})_t$  and  $\hat{\mathbf{A}}_j = (\hat{\mathbf{A}}_{j1}, \dots, \hat{\mathbf{A}}_{jH})'$ . Then for algorithm TSIR matrix

$$\widehat{\mathbf{M}}_{0j} = \widehat{\mathbf{Cov}}_{A_j}.$$

Denote  $\widehat{\mathbf{Cov}}_{jh}$  for a sample covariance matrix of  $\mathbf{Z}_{jh}$ . Then for algorithm TSAVE matrix

$$\widehat{\mathbf{M}}_{0j} = \frac{1}{H} \sum_{h=1}^H (\mathbf{I}_p - \widehat{\mathbf{Cov}}_{jh})^2.$$

For TSSH then matrix

$$\widehat{\mathbf{M}}_{2j} = a\widehat{\mathbf{M}}_{1j} + (1 - a)\widehat{\mathbf{M}}_{0j},$$

for a chosen  $0 \leq a \leq 1$ . Note that the value of  $H$  can be different for TSIR and TSAVE parts.

The algorithms find an orthogonal matrix  $\mathbf{U} = (\mathbf{u}_1, \dots, \mathbf{u}_p)'$  by maximizing, for  $b = 0, 1$  or  $2$ ,

$$\sum_{i \in k} \|\text{diag}(\mathbf{U}\widehat{\mathbf{M}}_{bj}\mathbf{U}')\|^2 = \sum_{i \in 1}^p \sum_{j \in k} (\mathbf{u}_i' \widehat{\mathbf{M}}_{bj} \mathbf{u}_i)^2.$$

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

Write  $\lambda_{ij} = c(\mathbf{u}_i' \widehat{\mathbf{M}}_{bj} \mathbf{u}_i)^2$ , where  $c$  is chosen in such way that  $\sum_{i=1}^p \sum_{j \in k} \lambda_{ij} = 1$ . Then the  $(i, j)$ :th element of the matrix  $\mathbf{L}$  is  $\lambda_{ij}$ .

To make a choice on which lags and directions to keep, see [summary.tssdr](#). Note that when printing a tssdr object, all elements are printed, except the directions  $\mathbf{S}$ .

### Value

A list with class 'tssdr' containing the following components:

w	The estimated signal separation matrix.
k	The vector of the used lags.

S	The estimated directions as time series object standardized to have mean 0 and unit variances.
L	The Lambda matrix for choosing lags and directions.
H	The used number of slices.
yname	The name for the response time series $y$ .
Xname	The name for the predictor time series $\mathbf{X}$ .
algorithm	The used algorithm as a character string.

**Author(s)**

Markus Matilainen

**References**

Matilainen M., Croux C., Nordhausen K. & Oja H. (2017), *Supervised Dimension Reduction for Multivariate Time Series*, *Econometrics and Statistics*, 4, 57–69.

Matilainen M., Croux C., Nordhausen K. & Oja H. *Sliced Average Variance Estimation for multivariate Time Series*. Submitted.

Li, K.C. (1991), *Sliced Inverse Regression for Dimension Reduction*, *Journal of the American Statistical Association*, 86, 316–327.

Cook, R., Weisberg, S. (1991), *Sliced Inverse Regression for Dimension Reduction*, *Comment. Journal of the American Statistical Association*, 86, 328–332.

**See Also**

[summary.tssdr](#), [dr](#)

**Examples**

```
n <- 10000
A <- matrix(rnorm(9), 3, 3)

x1 <- arima.sim(n = n, list(ar = 0.2))
x2 <- arima.sim(n = n, list(ar = 0.8))
x3 <- arima.sim(n = n, list(ar = 0.3, ma = -0.4))
eps2 <- rnorm(n - 1)
y <- 2*x1[1:(n - 1)] + eps2
X <- ((cbind(x1, x2, x3))[2:n, ]) %*% t(A)

res1 <- tssdr(y, X, algorithm = "TSAVE")
res1
summ1 <- summary(res1, type = "alllag", thres = 0.8)
summ1
plot(summ1)
head(components(summ1))
coef(summ1)

#Hybrid of TSIR and TSAVE. For TSIR part H = 10 and for TSAVE part H = 2.
tssdr(y, X, algorithm = "TSSH", weight = 0.6, H = c(10, 2))
```

## Description

The vSOBI method for the blind source separation problem. It is designed for time series with stochastic volatility. The method is a variant of SOBI, which is a method designed to separate ARMA sources, and an alternative to FixNA and FixNA2 methods.

## Usage

```
vSOBI(X, ...)

## Default S3 method:
vSOBI(X, k = 1:12, eps = 1e-06, maxiter = 1000, G = "pow",
      ordered = FALSE, acfk = NULL, original = TRUE, ...)
## S3 method for class 'ts'
vSOBI(X, ...)
```

## Arguments

X	A numeric matrix or a multivariate time series object of class <code>ts</code> . Missing values are not allowed.
k	A vector of lags. It can be any non-zero positive integer, or a vector consisting of them. Default is 1:12.
eps	Convergence tolerance.
maxiter	The maximum number of iterations.
G	Function $G(x)$ . The choices are <code>pow</code> (default) and <code>lcosh</code> .
ordered	Whether to order components according to their volatility. Default is <code>FALSE</code> .
acfk	A vector of lags to be used in testing the presence of serial autocorrelation. Applicable only if <code>ordered = TRUE</code> .
original	Whether to return the original components or their residuals based on ARMA fit. Default is <code>TRUE</code> , i.e. the original components are returned. Applicable only if <code>ordered = TRUE</code> .
...	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 algorithm finds an orthogonal matrix  $\mathbf{U}$  by maximizing

$$\mathbf{D}(\mathbf{U}) = \sum_{k=1}^K \mathbf{D}_k(\mathbf{U})$$

$$= \sum_{k=1}^K \sum_{i=1}^p \left( \frac{1}{T-k} \sum_{t=1}^{T-k} [G(\mathbf{u}'_i \mathbf{Y}_t) G(\mathbf{u}'_i \mathbf{Y}_{t+k})] - \left( \frac{1}{T-k} \right)^2 \sum_{t=1}^{T-k} [G(\mathbf{u}'_i \mathbf{Y}_t)] \sum_{t=1}^{T-k} [G(\mathbf{u}'_i \mathbf{Y}_{t+k})] \right)^2.$$

For function  $G(x)$  the choices are  $x^2$  and  $\log(\cosh(x))$ .

The algorithm works iteratively starting with  $\text{diag}(\rho)$  as an initial value for an orthogonal matrix  $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_p)'$ .

Matrix  $\mathbf{T}_{ik}$  is a partial derivative of  $\mathbf{D}_k(\mathbf{U})$  with respect to  $\mathbf{u}_i$ . Then  $\mathbf{T}_k = (\mathbf{T}_{1k}, \dots, \mathbf{T}_{pk})'$ , where  $p$  is the number of columns in  $\mathbf{Y}$ , and  $\mathbf{T} = \sum_{k=1}^K \mathbf{T}_k$ . The update for the orthogonal matrix  $\mathbf{U}_{new} = (\mathbf{T}\mathbf{T}')^{-1/2}\mathbf{T}$  is calculated at each iteration step. The algorithm stops when

$$\|\mathbf{U}_{new} - \mathbf{U}_{old}\|$$

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

For `ordered = TRUE` the function orders the sources according to their volatility. First a possible linear autocorrelation is removed using `auto.arima`. Then a squared autocorrelation test is performed for the sources (or for their residuals, when linear correlation is present). The sources are then put in a decreasing order according to the value of the test statistic of the squared autocorrelation test. For more information, see `lbtest`.

## Value

A list with class 'bssvol' (inherits from class 'bss') containing the following components:

<code>W</code>	The estimated unmixing matrix.
<code>k</code>	The vector of the used lags.
<code>S</code>	The estimated sources as time series object standardized to have mean 0 and unit variances. If <code>ordered = TRUE</code> , then components are ordered according to their volatility.

If `ordered = TRUE`, then also the following components included in the list:

<code>fits</code>	The ARMA fits for the components with linear autocorrelation.
<code>armaeff</code>	A logical vector. Has value 1 if ARMA fit was done to the corresponding component.
<code>linTS</code>	The value of the modified Ljung-Box test statistic for each component.
<code>linP</code>	P-value based on the modified Ljung-Box test statistic for each component.
<code>volTS</code>	The value of the volatility clustering test statistic.
<code>volP</code>	P-value based on the volatility clustering test statistic.

## Author(s)

Markus Matilainen

## References

- Belouchrani, A., Abed-Meriam, K., Cardoso, J.F. and Moulines, R. (1997), *A blind Source Separation Technique Using Second-Order Statistics*, IEEE Transactions on Signal Processing, 434–444.
- Matilainen, M., Miettinen, J., Nordhausen, K., Oja, H. and Taskinen, S. (2017), *On Independent Component Analysis with Stochastic Volatility Models*, Austrian Journal of Statistics, 46(3–4), 57–66.

**See Also**

[FixNA](#), [SOBI](#), [lbtest](#), [auto.arima](#)

**Examples**

```
library(stochvol)
n <- 10000
A <- matrix(rnorm(9), 3, 3)

# simulate SV models
s1 <- svsim(n, mu = -10, phi = 0.8, sigma = 0.1)$y
s2 <- svsim(n, mu = -10, phi = 0.9, sigma = 0.2)$y
s3 <- svsim(n, mu = -10, phi = 0.95, sigma = 0.4)$y

# create a daily time series
X <- ts(cbind(s1, s2, s3) %*% t(A), end = c(2015, 338), frequency = 365.25)

res <- vSOBI(X)
res
coef(res)
plot(res)
head(bss.components(res))

MD(res$W, A) # Minimum Distance Index, should be close to zero
```

---

WeeklyReturnsData	<i>Logarithmic Returns of Exchange Rates of 7 Currencies Against US Dollar</i>
-------------------	--

---

**Description**

This data set has logarithmic returns of exchange rates of 7 currencies against US dollar extracted from the International Monetary Fund's (IMF) database. These currencies are Australian Dollar (AUD), Canadian Dollar (CAD), Norwegian Kroner (NOK), Singapore Dollar (SGD), Swedish Kroner (SEK), Swiss Franc (CHF) and British Pound (GBP).

**Usage**

```
data("WeeklyReturnsData")
```

**Format**

An object of class `ts` with 605 observations on the following 7 variables.

AUD The weekly logarithmic returns  $r_{AUD,t}$  of the exchange rates of AUD against US Dollar.

CAD The weekly logarithmic returns  $r_{CAD,t}$  of the exchange rates of CAD against US Dollar.

NOK The weekly logarithmic returns  $r_{NOK,t}$  of the exchange rates of NOK against US Dollar.

SGD The weekly logarithmic returns  $r_{SGD,t}$  of the exchange rates of SGD against US Dollar.

SEK The weekly logarithmic returns  $r_{SEK,t}$  of the exchange rates of SEK against US Dollar.

CHF The weekly logarithmic returns  $r_{CHF,t}$  of the exchange rates of CHF against US Dollar.

GBP The weekly logarithmic returns  $r_{GBP,t}$  of the exchange rates of GBP against US Dollar.

## Details

The daily exchange rates of the currencies against US Dollar from March 22, 2000 to October 26, 2011 are extracted from the International Monetary Fund's (IMF) Exchange Rates database from <http://www.imf.org/external/np/fin/ert/GUI/Pages/CountryDataBase.aspx>. These rates are representative rates (currency units per US Dollar), which are reported daily to the IMF by the issuing central bank.

The weekly averages of these exchange rates are then calculated. The logarithmic returns of the average weekly exchange rates are calculated for the currency  $j$  as follows.

Let  $x_{j,t}$  be the exchange rates of  $j$  against US Dollar. Then

$$r_{j,t} = \log(x_{j,t}) - \log(x_{j,t-1}),$$

where  $t = 1, \dots, 605$  and  $j = AUD, CAD, NOK, SGD, SEK, CHF, GBP$ . The six missing values in  $r_{SEK,t}$  are changed to 0. The assumption used here is that there has not been any change from the previous week.

The weekly returns data is then changed to a multivariate time series object of class `ts`. The resulting `ts` object is then dataset `WeeklyReturnsData`.

An example analysis of the data is given in Miettinen et al. (2017). Same data has also been used in Hu and Tsay (2014).

## Source

International Monetary Fund (2017), *IMF Exchange Rates*, <http://www.imf.org/external/np/fin/ert/GUI/Pages/CountryDataBase.aspx>

For IMF Copyrights and Usage, and special terms and conditions pertaining to the use of IMF data, see <http://www.imf.org/external/terms.htm>

## References

Miettinen, M., Matilainen, M., Nordhausen, K. and Taskinen, S. (2017), *Extracting Conditionally Heteroscedastic Components Using ICA*, Submitted.

Hu and Tsay (2014), *Principal Volatility Component Analysis*, *Journal of Business & Economic Statistics*, 32(2), 153–164.

## Examples

```
plot(WeeklyReturnsData)

res <- gSOBI(WeeklyReturnsData)
res
```



```
coef(res)  
plot(res)  
head(bss.components(res))
```

# Index

## \*Topic **datasets**

WeeklyReturnsData, 31

## \*Topic **methods**

coef.summary.tssdr, 3  
components.summary.tssdr, 4  
components.tssdr, 5  
plot.summary.tssdr, 18  
plot.tssdr, 19  
print.bssvol, 19  
print.lbttest, 20  
print.summary.tssdr, 20  
print.tssdr, 21  
summary.tssdr, 24

## \*Topic **multivariate**

FixNA, 5  
gFOBI, 8  
gJADE, 11  
PVC, 22  
tsBSS-package, 2  
tssdr, 26  
vSOBI, 29

## \*Topic **package**

tsBSS-package, 2

## \*Topic **ts**

FixNA, 5  
gFOBI, 8  
gJADE, 11  
PVC, 22  
tsBSS-package, 2  
tssdr, 26  
vSOBI, 29

auto.arima, 7, 9, 10, 12, 13, 15, 16, 23, 30, 31

coef.summary.tssdr, 3  
components.summary.tssdr, 4  
components.tssdr, 5  
comVol, 23

dr, 3, 28

FixNA, 2, 3, 5, 17, 31

FOBI, 2, 8, 10

frjd, 9–11, 13, 27

gFOBI, 2, 8, 12, 13, 17

gJADE, 3, 11, 17

gSOBI, 13, 17, 23

JADE, 3, 11, 13

lbttest, 7, 9, 10, 12, 13, 15, 16, 16, 20, 23, 30, 31

plot.summary.tssdr, 18

plot.ts, 18, 19

plot.tssdr, 19

print.bss, 20

print.bssvol, 19

print.lbttest, 17, 20

print.summary.tssdr, 20

print.tssdr, 21

PVC, 22

rjd, 9, 11, 27

SOBI, 3, 16, 31

summary.tssdr, 4, 18, 21, 24, 27, 28

ts, 6, 8, 11, 14, 16, 22, 26, 29, 32

tsBSS-package, 2

tssdr, 3, 5, 19, 21, 25, 26

vSOBI, 2, 3, 7, 16, 17, 29

WeeklyReturnsData, 3, 31