Package ‘tsBSS’

October 14, 2019

Type Package

Title Blind Source Separation and Supervised Dimension Reduction for Time Series

Version 0.5.5

Date 2019-10-14

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

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

Depends ICtest (>= 0.3-2), JADE (>= 2.0-2)

Imports Rcpp (>= 0.11.0), forecast, boot, parallel, xts, zoo

LinkingTo Rcpp, RcppArmadillo

Suggests stochvol

Description Different estimators 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>). Different functions based on AMUSE and SOBI are also provided for estimating the dimension of the white noise subspace.

LazyData true

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-10-14 08:40:02 UTC

R topics documented:

`tsBSS-package` *Blind Source Separation and Supervised Dimension Reduction for Time Series*

Description

Different estimators 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>). Different functions based on AMUSE and SOBI are also provided for estimating the dimension of the white noise subspace.

Details

Package:	tsBSS
Type:	Package
Version:	0.5.5
Date:	2019-10-14
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 acommodate 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 acommodate stochastic volatility.
- `gSOBI` Function to solve the BSS problem. Algorithm is a combination of `SOBI` and `vSOBI` algorithms.
- `PVC` Function to solve the BSS problem. Algorithm is a modified version of Principal Component Volatility Analysis by Hu and Tsay (2011).

The input data can be a numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. For other classes, the `tsbox` package provides appropriate conversions to and from these classes.

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.

Several functions in this package utilize `"rjd"` (real joint diagonalization) and `"frjd"` (fast rjd) from the `JADE` package for joint diagonalization of k real-valued square matrices.

There are several functions for estimating the number of white noise latent series in second-order source separation (SOS) models. The functions are

- `AMUSEboot`, `AMUSEladle` and `AMUSEasymp` which are based on `AMUSE`.
- `SOBIboot`, `SOBIladle` and `SOBIasymp` which are based on `SOBI`.

Additionally, there is function `lbtest` for a modified Ljung-Box test and a volatility clustering test for univariate and multivariate time series.

The package also contains a dataset `WeeklyReturnsData`, which 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, Joni Virta

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

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. and Oja H. (2017), *Supervised Dimension Reduction for Multivariate Time Series*, Econometrics and Statistics, 4, 57–69.

Matilainen M., Croux C., Nordhausen K. and Oja H. (2019), *Sliced Average Variance Estimation for Multivariate Time Series*. Statistics: A Journal of Theoretical and Applied Statistics, 53, 630–655.

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.

Matilainen, M., Nordhausen, K. and Virta, J. (2018), *On the Number of Signals in Multivariate Time Series*. In Deville, Y., Gannot, S., Mason, R., Plumbley, M.D. and Ward, D. (editors) "International Conference on Latent Variable Analysis and Signal Separation", LNCS 10891, 248–258. Springer, Cham., <doi:10.1007/978-3-319-93764-9_24>.

Nordhausen, K. and Virta, J.(2018), *Ladle Estimator for Time Series Signal Dimension*. In 2018 IEEE Statistical Signal Processing Workshop (SSP), pp. 428–432, <doi:10.1109/SSP.2018.8450695>.

Virta, J. and Nordhausen, K. (2019), *Determining the Signal Dimension in Second Order Source Separation*. To appear in Statistica Sinica, <doi:10.5705/ss.202018.0347>.

Miettinen, M., Matilainen, M., Nordhausen, K. and Taskinen, S. (2019), *Extracting Conditionally Heteroskedastic Components Using Independent Component Analysis*. Accepted for publication in Journal of Time Series Analysis. Available at http://arxiv.org/abs/1811.10963v1.

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

`AMUSEasymp` *Second-order Separation Sub-White-Noise Asymptotic Testing with AMUSE*

Description

The function uses AMUSE (Algorithm for Multiple Unknown Signals Extraction) to test whether the last `p-k` latent series are pure white noise, assuming a p-variate second-order stationary blind source separation (BSS) model. The test is asymptotic.

Usage

```
AMUSEasymp(X, ...)

## Default S3 method:
AMUSEasymp(X, k, tau = 1, ...)
## S3 method for class 'ts'
AMUSEasymp(X, ...)
## S3 method for class 'xts'
AMUSEasymp(X, ...)
## S3 method for class 'zoo'
AMUSEasymp(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. Missing values are not allowed.
`k`	The number of latent series that are not white noise. Can be between 0 and $p-1$.
`tau`	The lag for the AMUSE autocovariance matrix.
`...`	Further arguments to be passed to or from methods.

Details

AMUSE standardizes `X` with n samples and computes the eigenedcomposition of the autocovariance matrix of the standardized data for a chosen lag `tau`, yielding a transformation $\mathbf{W}$ giving the latent variables as $\mathbf{S} = \mathbf{X}\mathbf{W}$. Assume, without loss of generality, that the latent components are ordered in decreasing order with respect to the squares of the corresponding eigenvalues of the autocovariance matrix. Under the null hypothesis the final $p-k$ eigenvalues equal zero, $\lambda_{p-k} = \cdots = \lambda_p = 0$, and their mean square m can be used as a test statistic in inference on the true number of latent white noise series.

This function conducts the hypothesis test using the asymptotic null distribution of m, a chi-squared distribution with $(p-k)(p-k+1)/2$ degrees of freedom.

Value

A list of class ictest inheriting from class htest containing:

`statistic`	The value of the test statistic.
`p.value`	The p-value of the test.
`parameter`	The degrees of freedom of the asymptotic null distribution.
`method`	Character string indicating which test was performed.
`data.name`	Character string giving the name of the data.
`alternative`	Character string specifying the alternative hypothesis.
`k`	The number of latent series that are not white noise used in the testing problem.
`W`	The transformation matrix to the latent series.
`S`	Multivariate time series with the centered source components.
`D`	The underlying eigenvalues of the autocovariance matrix.
`MU`	The location of the data which was subtracted before calculating AMUSE.
`tau`	The used lag.
`...`	Further arguments to be passed to or from methods.

Author(s)

Klaus Nordhausen, Joni Virta

References

Virta, J. and Nordhausen, K. (2019), *Determining the Signal Dimension in Second Order Source Separation*. To appear in Statistica Sinica, <doi:10.5705/ss.202018.0347>.

See Also

AMUSE, SOBI, SOBIasymp

Examples

```
n <- 1000

A <- matrix(rnorm(16), 4, 4)
s1 <- arima.sim(list(ar = c(0.3, 0.6)), n)
s2 <- arima.sim(list(ma = c(-0.3, 0.3)), n)
s3 <- rnorm(n)
s4 <- rnorm(n)

S <- cbind(s1, s2, s3, s4)
X <- S %*% t(A)

asymp_res_1 <- AMUSEasymp(X, k = 1)
asymp_res_1

asymp_res_2 <- AMUSEasymp(X, k = 2)
asymp_res_2

# Plots of the estimated sources, the last two are white noise
plot(asymp_res_2)
```

AMUSEboot — *Second-order Separation Sub-White-Noise Bootstrap Testing with AMUSE*

Description

The function uses AMUSE (Algorithm for Multiple Unknown Signals Extraction) to test whether the last `p-k` latent series are pure white noise, assuming a p-variate second-order stationary blind source separation (BSS) model. Four different bootstrapping strategies are available and the function can be run in parallel.

Usage

```
AMUSEboot(X, ...)

## Default S3 method:
AMUSEboot(X, k, tau = 1, n.boot = 200, s.boot = c("p", "np1", "np2", "np3"),
          ncores = NULL, iseed = NULL, ...)
## S3 method for class 'ts'
AMUSEboot(X, ...)
## S3 method for class 'xts'
AMUSEboot(X, ...)
## S3 method for class 'zoo'
AMUSEboot(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. Missing values are not allowed.
`k`	The number of latent series that are not white noise. Can be between 0 and $p-1$.
`tau`	The lag for the AMUSE autocovariance matrix.
`n.boot`	The number of bootstrapping samples.
`s.boot`	Bootstrapping strategy to be used. Possible values are `"p"` (default), `"np1"`, `"np2"`, `"np3"`. See details for further information.
`ncores`	The number of cores to be used. If `NULL` or 1, no parallel computing is used. Otherwise `makeCluster` with `type = "PSOCK"` is used. It is the users repsonsibilty to choose a reasonable value for `ncores`. The function `detectCores` might be useful in this context.
`iseed`	If parallel computation is used, the seed passed on to `clusterSetRNGStream`. Default is `NULL` which means no fixed seed is used.
`...`	Further arguments to be passed to or from methods.

Details

AMUSE standardizes `X` with n samples and computes the eigendecomposition of the autocovariance matrix of the standardized data for a chosen lag `tau`, yielding a transformation $\mathbf{W}$ giving the latent variables as $\mathbf{S} = \mathbf{XW}$. Assume, without loss of generality, that the latent components are ordered in decreasing order with respect to the squares of the corresponding eigenvalues of the autocovariance matrix. Under the null hypothesis the final $p-k$ eigenvalues equal zero, $\lambda_{p-k} = \cdots = \lambda_p = 0$, and their mean square m can be used as a test statistic in bootstrap-based inference on the true number of latent white noise series.

The function offers four different bootstrapping strategies for generating samples for which the null hypothesis approximately holds, and they are all based on the following general formula:

1. Decompose the AMUSE-estimated latent series $\mathbf{S}$ into the postulated signal $\mathbf{S}_1$ and white noise $\mathbf{S}_2$.
2. Take n bootstrap samples $\mathbf{S}_2^*$ of $\mathbf{S}_2$, see the different strategies below.
3. Recombine $\mathbf{S}^* = (\mathbf{S_1}, \mathbf{S_2^*})$ and back-transform $\mathbf{X}^* = \mathbf{S}^*\mathbf{W}^{-1}$.
4. Compute the test statistic based on $\mathbf{X}^*$.
5. Repeat the previous steps `n.boot` times.

The four different bootstrapping strategies are:

1. `s.boot = "p"`: The first strategy is parametric and simply generates all boostrap samples independently and identically from the standard normal distribution.
2. `s.boot = "np1"`: The second strategy is non-parametric and pools all observed $n(p-k)$ white noise observations together and draws the bootstrap samples from amongst them.
3. `s.boot = "np2"`: The third strategy is non-parametric and proceeds otherwise as the second strategy but acts component-wise. That is, for each of the $p-k$ white noise series it pools the observed n white noise observations together and draws the bootstrap samples of that particular latent series from amongst them.

4. `s.boot = "np3"`: The third strategy is non-parametric and instead of drawing the samples univariately as in the second and third strategies, it proceeds by resampling n vectors of size $p - k$ from amongst all the observed n white noise vectors.

The function can be run in parallel by setting `ncores` to the desired number of cores (should be less than the number of cores available - 1). When running code in parallel the standard random seed of R is overridden and if a random seed needs to be set it should be passed via the argument `iseed`. The argument `iseed` has no effect in case `ncores` equals 1 (the default value).

Value

A list of class ictest inheriting from class htest containing:

`statistic`	The value of the test statistic.
`p.value`	The p-value of the test.
`parameter`	The number of bootstrap samples.
`alternative`	Character string specifying the alternative hypothesis.
`k`	The number of latent series that are not white noise used in the testing problem.
`W`	The transformation matrix to the latent series.
`S`	Multivariate time series with the centered source components.
`D`	The underlying eigenvalues of the autocovariance matrix.
`MU`	The location of the data which was subtracted before calculating AMUSE.
`tau`	The used lag.
`method`	Character string indicating which test was performed.
`data.name`	Character string giving the name of the data.
`s.boot`	Character string denoting which bootstrapping test version was used.

Author(s)

Markus Matilainen, Klaus Nordhausen, Joni Virta

References

Matilainen, M., Nordhausen, K. and Virta, J. (2018), *On the Number of Signals in Multivariate Time Series*. In Deville, Y., Gannot, S., Mason, R., Plumbley, M.D. and Ward, D. (editors) "International Conference on Latent Variable Analysis and Signal Separation", LNCS 10891, 248–258. Springer, Cham., <doi:10.1007/978-3-319-93764-9_24>.

See Also

`AMUSE`, `SOBI`, `SOBIboot`

Examples

```
n <- 1000

A <- matrix(rnorm(16), 4, 4)
s1 <- arima.sim(list(ar = c(0.3, 0.6)), n)
s2 <- arima.sim(list(ma = c(-0.3, 0.3)), n)
s3 <- rnorm(n)
s4 <- rnorm(n)

S <- cbind(s1, s2, s3, s4)
X <- S %*% t(A)

boot_res_1 <- AMUSEboot(X, k = 1)
boot_res_1

boot_res_2 <- AMUSEboot(X, k = 2)
boot_res_2

# Plots of the estimated sources, the last two are white noise
plot(boot_res_2)
```

AMUSEladle — *Ladle Estimator to Estimate the Number of White Noise Components in SOS with AMUSE*

Description

The ladle estimator uses the eigenvalues and eigenvectors of an autocovariance matrix with the chosen lag to estimate the number of white noise components in second-order source separation (SOS).

Usage

```
AMUSEladle(X, ...)

## Default S3 method:
AMUSEladle(X, tau = 1, l = 20, sim = c("geom", "fixed"), n.boot = 200,
ncomp = ifelse(ncol(X) > 10, floor(ncol(X)/log(ncol(X))), ncol(X) - 1), ...)
## S3 method for class 'ts'
AMUSEladle(X, ...)
## S3 method for class 'xts'
AMUSEladle(X, ...)
## S3 method for class 'zoo'
AMUSEladle(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. Missing values are not allowed.
`tau`	The lag for the AMUSE autocovariance matrix.
`l`	If `sim = "geom"` (default) then `l` is the success probability of the geometric distribution from where the bootstrap block lengths for the stationary bootstrap are drawn. If `sim = "fixed"` then `l` is the fixed block length for the fixed block bootstrap.
`sim`	If `"geom"` (default) then the stationary bootstrap is used. If `"fixed"` then the fixed block bootstrap is used.
`n.boot`	The number of bootstrapping samples. See `tsboot` for details.
`ncomp`	The number of components among which the ladle estimator is to be searched. Must be between `0` and `ncol(X)-1`. The default follows the recommendation of Luo and Li (2016).
`...`	Further arguments to be passed to or from methods.

Details

AMUSE standardizes `X` with n samples and computes the eigenedcomposition of the autocovariance matrix of the standardized data for a chosen lag `tau`, yielding a transformation $\mathbf{W}$ giving the latent variables as $\mathbf{S} = \mathbf{X}\mathbf{W}$. Assume, without loss of generality, that the latent components are ordered in decreasing order with respect to the squares of the corresponding eigenvalues of the autocovariance matrix. Under the assumption that we have k non-white-noise components, the final $p - k$ eigenvalues equal zero, $\lambda_{p-k} = \cdots = \lambda_p = 0$.

The change point from non-zero eigenvalues to zero eigenvalues is visible in the eigenvectors of the autocovariance matrix as an increase in their boostrap variablity. Similarly, before the change point, the squared eigenvalues decrease in magnitude and afterwards they stay constant. The ladle estimate combines the scaled eigenvector bootstrap variability with the scaled eigenvalues to estimate the number of non-white-noise components. The estimate is the value of $k = 0, \ldots,$ `ncomp` where the combined measure achieves its minimum value.

Value

A list of class `ladle` containing:

`method`	The string AMUSE.
`k`	The estimated number of non-white-noise components.
`fn`	The vector giving the measures of variation of the eigenvectors using the bootstrapped eigenvectors for the different number of components.
`phin`	Normalized eigenvalues of the AMUSE matrix.
`data.name`	The name of the data for which the ladle estimate was computed.
`gn`	The main criterion for the ladle estimate - the sum of `fn` and `phin`. `k` is the value where `gn` takes its minimum.
`lambda`	The eigenvalues of the AMUSE matrix.

`W`	The transformation matrix to the source components. Also known as the unmixing matrix.
`S`	Multivariate time series with the centered source components.
`MU`	The location of the data which was subtracted before calculating the source components.
`sim`	The used boostrapping technique, either `"geom"` or `"fixed"`.
`lag`	The used lag.

Author(s)

Klaus Nordhausen, Joni Virta

References

Nordhausen, K. and Virta, J.(2018), *Ladle Estimator for Time Series Signal Dimension*. In 2018 IEEE Statistical Signal Processing Workshop (SSP), pp. 428–432, <doi:10.1109/SSP.2018.8450695>.

Luo, W. and Li, B. (2016), *Combining Eigenvalues and Variation of Eigenvectors for Order Determination*, Biometrika, 103. 875–887. <doi:10.1093/biomet/asw051>

See Also

`AMUSE`, `SOBI`, `SOBIladle`

Examples

```
n <- 1000

s1 <- arima.sim(n = n, list(ar = 0.6, ma = c(0, -0.4)))
s2 <- arima.sim(n = n, list(ar = c(0.4,0.1,0.3), ma = c(0.2, 0.4)))
s3 <- arima.sim(n = n, list(ar = c(0.7, 0.1)))
Snoise <- matrix(rnorm(5*n), ncol = 5)
S <- cbind(s1, s2, s3, Snoise)

A <- matrix(rnorm(64), 8, 8)
X <- S %*% t(A)

ladle_AMUSE <- AMUSEladle(X, l = 20, sim = "geom")

# The estimated number of non-white-noise components
summary(ladle_AMUSE)

# The ladle plot
ladleplot(ladle_AMUSE)
# Using ggplot
ggladleplot(ladle_AMUSE)

# Time series plots of the estimated components
plot(ladle_AMUSE)
```

`bssvol` *Class: bssvol*

Description

Class bssvol (blind source separation in stochastic volatility processes) with methods print.bssvol (prints an object of class bssvol) and plot.bss (plots an object of class bssvol).

Class also inherits methods from the class bss in package `JADE`: for extracting the components of an object of class bssvol (`bss.components`) and the coefficients of an object of class bssvol (`coef.bss`).

Usage

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

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

Arguments

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

Author(s)

Markus Matilainen

See Also

`JADE`, `bss.components`, `coef.bss`

`FixNA` *The FixNA method for Blind Source Separation*

Description

The FixNA (Fixed-point algorithm for maximizing the Nonlinear Autocorrelation; Shi et al., 2009) and FixNA2 (Matilainen et al., 2017) methods for blind source separation of 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 = c("pow", "lcosh"),
                        method = c("FixNA", "FixNA2"),
                  ordered = FALSE, acfk = NULL, original = TRUE, alpha = 0.05, ...)
## S3 method for class 'ts'
FixNA(X, ...)
## S3 method for class 'xts'
FixNA(X, ...)
## S3 method for class 'zoo'
FixNA(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. 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 `"pow"` (default) and `"lcosh"`.
`method`	The method to be used. The choices are `"FixNA"` (default) and `"FixNA2"`.
`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`.
`alpha`	Alpha level for linear correlation detection. Default is 0.05.
`...`	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

$$\mathbf{D}_2(\mathbf{U}) = \sum_{k=1}^{K}\mathbf{D}_{2k}(\mathbf{U})$$

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

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

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

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 $\mathbf{T}_{mk} = (\mathbf{T}_{m1k}, \ldots, \mathbf{T}_{mpk})'$, 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' containing the following components:

`W`	The estimated unmixing matrix. If `ordered = TRUE`, the rows are ordered according to the order of the components.
`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 `original = FALSE`, the sources with linear autocorrelation are replaced by their ARMA residuals.
`MU`	The means of the original series.

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

`Sraw`	The ordered original estimated sources as time series object standardized to have mean 0 and unit variances. Returned only if `original = FALSE`.
`fits`	The ARMA fits for the components with linear autocorrelation.
`armaeff`	A logical vector. Is TRUE 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

References

Hyvärinen, 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 (generalized Fourth Order Blind Identification) method for blind source separation of 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 = c("frjd", "rjd"),
                        na.action = na.fail, weight = NULL, ordered = FALSE,
                        acfk = NULL, original = TRUE, alpha = 0.05, ...)
## S3 method for class 'ts'
gFOBI(X, ...)
## S3 method for class 'xts'
gFOBI(X, ...)
## S3 method for class 'zoo'
gFOBI(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. 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 `"rjd"` and `"frjd"`. Default is `"frjd"`.
`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`.
`alpha`	Alpha level for linear correlation detection. Default is 0.05.
`...`	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 $\mathbf{X}$. 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

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

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 `lbtest`.

Value

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

`W`	The estimated unmixing matrix. If `ordered = TRUE`, the rows are ordered according to the order of the components.
`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 `original = FALSE`, the sources with linear autocorrelation are replaced by their ARMA residuals.
`MU`	The means of the original series.

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

`Sraw`	The ordered original estimated sources as time series object standardized to have mean 0 and unit variances. Returned only if `original = FALSE`.
`fits`	The ARMA fits for the components with linear autocorrelation.
`armaeff`	A logical vector. Is TRUE 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, 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 <- 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 (generalized Joint Approximate Diagonalization of Eigenmatrices) method for blind source separation of 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 = c("frjd", "rjd"),
                           na.action = na.fail, weight = NULL, ordered = FALSE,
                           acfk = NULL, original = TRUE, alpha = 0.05, ...)
```

```
## S3 method for class 'ts'
gJADE(X, ...)
## S3 method for class 'xts'
gJADE(X, ...)
## S3 method for class 'zoo'
gJADE(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. 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 `JADE`.
`eps`	Convergence tolerance.
`maxiter`	The maximum number of iterations.
`method`	The method to use for the joint diagonalization. The options are `"rjd"` and `"frjd"`. Default is `"frjd"`.
`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`.
`alpha`	Alpha level for linear correlation detection. Default is 0.05.
`...`	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 matrix $\widehat{\mathbf{C}}_k^{ij}(\mathbf{Y})$ is of the form

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

for $i, j = 1, \ldots, p$, where $\mathbf{S}_k(\mathbf{Y})$ is the lagged sample covariance matrix of $\mathbf{Y}$ for lag $k = 1, \ldots, K$, $\mathbf{E}^{ij}$ 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

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

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' containing the following components:

`W`	The estimated unmixing matrix. If `ordered = TRUE`, the rows are ordered according to the order of the components.
`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 `original = FALSE`, the sources with linear autocorrelation are replaced by their ARMA residuals.
`MU`	The means of the original series.

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

`Sraw`	The ordered original estimated sources as time series object standardized to have mean 0 and unit variances. Returned only if `original = FALSE`.
`fits`	The ARMA fits for the components with linear autocorrelation.
`armaeff`	A logical vector. Is TRUE 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)

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 (generalized Second Order Blind Identification) method for the blind source separation (BSS) problem. The method is designed for separating multivariate time series 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, ...)

## Default S3 method:
gSOBI(X, k1 = 1:12, k2 = 1:3, b = 0.9, eps = 1e-06, maxiter = 1000, ordered = FALSE,
      acfk = NULL, original = TRUE, alpha = 0.05, ...)
## S3 method for class 'ts'
gSOBI(X, ...)
## S3 method for class 'xts'
gSOBI(X, ...)
## S3 method for class 'zoo'
gSOBI(X, ...)
```

Arguments

X	A numeric matrix or a multivariate time series object of class ts, xts or zoo. 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`.
alpha	Alpha level for linear correlation detection. Default is 0.05.
...	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

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

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 `diag(p)` as an initial value for an orthogonal matrix $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p)'$.

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 $\mathbf{T}_{kj} = (\mathbf{T}_{1kj}, \ldots, \mathbf{T}_{pkj})'$, 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 $\mathbf{W} = \mathbf{US}^{-1/2}$.

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' containing the following components:

W	The estimated unmixing matrix. If ordered = TRUE, the rows are ordered according to the order of the components.
k1	The vector of the used lags for the SOBI part.
k2	The vector of the used lags for the vSOBI part.
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 original = FALSE, the sources with linear autocorrelation are replaced by their ARMA residuals.
MU	The means of the original series.

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

Sraw	The ordered original estimated sources as time series object standardized to have mean 0 and unit variances. Returned only if original = FALSE.
fits	The ARMA fits for the components with linear autocorrelation.
armaeff	A logical vector. Is TRUE 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, 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. (2019), *Extracting Conditionally Heteroskedastic Components Using Independent Component Analysis*. Accepted for publication in Journal of Time Series Analysis. Available at http://arxiv.org/abs/1811.10963v1.

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

# xts series as input
library(xts)
data(sample_matrix)
X2 <- as.xts(sample_matrix)
res2 <- gSOBI(X2, 1:4, 1:2, 0.99)
plot(res2, multi.panel = TRUE)

# zoo series as input
X3 <- as.zoo(X)
res3 <- gSOBI(X3, 1:4, 1:2, 0.99)
plot(res3)
```

---

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

Description

Modified Ljung-Box test and volatility clustering test for time series. Time series can be univariate or multivariate. 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 = c("squared", "linear"))

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

Arguments

`X`	A numeric vector/matrix or a univariate/multivariate time series object of class `ts`, `xts` or `zoo`. 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"`. In methods for class 'lbtest' only:
`x`	An object of class lbtest
`digits`	The number of digits when printing an object of class lbtest. Default is 3
`...`	Further arguments to be passed to or from methods.

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 χ^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. (2019), *Extracting Conditionally Heteroskedastic Components Using Independent Component Analysis*. Accepted for publication in Journal of Time Series Analysis. Available at http://arxiv.org/abs/1811.10963v1.

See Also

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)

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

PVC — *A modified algorithm for Principal Volatility Component estimator*

Description

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

Usage

```
PVC(X, ...)

## Default S3 method:
PVC(X, k = 1:12, ordered = FALSE, acfk = NULL, original = TRUE, alpha = 0.05, ...)
## S3 method for class 'ts'
PVC(X, ...)
## S3 method for class 'xts'
PVC(X, ...)
## S3 method for class 'zoo'
PVC(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. 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`.
`alpha`	Alpha level for linear correlation detection. Default is 0.05.
`...`	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 $\mathbf{X}$. 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 $Y_{ij,t-k} = Y_{i,t-k}Y_{j,t-k}, i, j = 1, \ldots, p$. 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 $k = 1, \ldots, K$ 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' containing the following components:

- `W` The estimated unmixing matrix. If `ordered = TRUE`, the rows are ordered according to the order of the components.
- `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 `original = FALSE`, the sources with linear autocorrelation are replaced by their ARMA residuals.
- `MU` The means of the original series.

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

- `Sraw` The ordered original estimated sources as time series object standardized to have mean 0 and unit variances. Returned only if `original = FALSE`.
- `fits` The ARMA fits for the components with linear autocorrelation.
- `armaeff` A logical vector. Is TRUE 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)

Jari Miettinen, Markus Matilainen

References

Miettinen, M., Matilainen, M., Nordhausen, K. and Taskinen, S. (2019), *Extracting Conditionally Heteroskedastic Components Using Independent Component Analysis*. Accepted for publication in Journal of Time Series Analysis. Available at http://arxiv.org/abs/1811.10963v1.

Hu, Y.-P. and 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
```

SOBIasymp — *Second-order Separation Sub-White-Noise Asymptotic Testing with SOBI*

Description

The function uses SOBI (Second Order Blind Identification) to test whether the last p-k latent series are pure white noise, assuming a p-variate second-order stationary blind source separation (BSS) model. The test is asymptotic.

Usage

```
SOBIasymp(X, ...)

## Default S3 method:
SOBIasymp(X, k, tau = 1:12, eps = 1e-06, maxiter = 200, ...)
## S3 method for class 'ts'
SOBIasymp(X, ...)
## S3 method for class 'xts'
SOBIasymp(X, ...)
## S3 method for class 'zoo'
SOBIasymp(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. Missing values are not allowed.
`k`	The number of latent series that are not white noise. Can be between 0 and $p-1$.
`tau`	The lags for the SOBI autocovariance matrices.
`eps`	The convergence tolerance for the joint diagonalization.
`maxiter`	The maximum number of iterations for the joint diagonalization.
`...`	Further arguments to be passed to or from methods.

Details

SOBI standardizes `X` with n samples and jointly diagonalizes the autocovariance matrices of the standardized data for a chosen set of lags `tau`, yielding a transformation $\mathbf{W}$ giving the latent variables as $\mathbf{S} = \mathbf{XW}$. Assume, without loss of generality, that the latent components are ordered in decreasing order with respect to the sums of squares of the corresponding "eigenvalues" produced by the joint diagonalization. Under the null hypothesis the lower right corner $(p-k) \times (p-k)$ blocks of the autocovariance matrices of the sources are zero matrices and the sum m of their squared norms over all lags can be used as a test statistic in inference on the true number of latent white noise series.

This function conducts the hypothesis test using the asymptotic null distribution of m, a chi-squared distribution with $T(p-k)(p-k+1)/2$ degrees of freedom where T is the number of autocovariance matrices used by SOBI.

Value

A list of class ictest inheriting from class htest containing:

`statistic`	The value of the test statistic.
`p.value`	The p-value of the test.
`parameter`	The degrees of freedom of the asymptotic null distribution.
`method`	Character string indicating which test was performed.
`data.name`	Character string giving the name of the data.
`alternative`	Character string specifying the alternative hypothesis.
`k`	The number of latent series that are not white noise used in the testing problem.
`W`	The transformation matrix to the latent series.
`S`	Multivariate time series with the centered source components.
`D`	The underlying eigenvalues of the autocovariance matrix.
`MU`	The location of the data which was subtracted before calculating SOBI.
`tau`	The used set of lags for the SOBI autocovariance matrices.

Author(s)

Klaus Nordhausen, Joni Virta

References

Virta, J. and Nordhausen, K. (2019), *Determining the Signal Dimension in Second Order Source Separation*. To appear in Statistica Sinica, <doi:10.5705/ss.202018.0347>.

See Also

AMUSE, SOBI, AMUSEasymp

Examples

```
n <- 1000

A <- matrix(rnorm(16), 4, 4)
s1 <- arima.sim(list(ar = c(0, 0.6)), n)
s2 <- arima.sim(list(ma = c(0, -0.5)), n)
s3 <- rnorm(n)
s4 <- rnorm(n)

S <- cbind(s1, s2, s3, s4)
X <- S %*% t(A)

asymp_res_1 <- SOBIasymp(X, k = 1)
asymp_res_1

asymp_res_2 <- SOBIasymp(X, k = 2)
asymp_res_2

# Plots of the estimated sources, the last two are white noise
plot(asymp_res_2)

# Note that AMUSEasymp with lag 1 does not work due to the lack of short range dependencies
AMUSEasymp(X, k = 1)
```

SOBIboot *Second-order Separation Sub-White-Noise Bootstrap Testing with SOBI*

Description

The function uses SOBI (Second Order Blind Identification) to test whether the last p-k latent series are pure white noise, assuming a p-variate second-order stationary blind source separation (BSS) model. Four different bootstrapping strategies are available and the function can be run in parallel.

Usage

```
SOBIboot(X, ...)

## Default S3 method:
SOBIboot(X, k, tau = 1:12, n.boot = 200, s.boot = c("p", "np1", "np2", "np3"),
```

```
  ncores = NULL, iseed = NULL, eps = 1e-06, maxiter = 200, ...)
## S3 method for class 'ts'
SOBIboot(X, ...)
## S3 method for class 'xts'
SOBIboot(X, ...)
## S3 method for class 'zoo'
SOBIboot(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. Missing values are not allowed.
`k`	The number of latent series that are not white noise. Can be between 0 and $p-1$.
`tau`	The vector of lags for the SOBI autocovariance matrices.
`n.boot`	The number of bootstrapping samples.
`s.boot`	Bootstrapping strategy to be used. Possible values are `"p"` (default), `"np1"`, `"np2"`, `"np3"`. See details for further information.
`ncores`	The number of cores to be used. If `NULL` or 1, no parallel computing is used. Otherwise `makeCluster` with `type = "PSOCK"` is used. It is the users repsonsibilty to choose a reasonable value for `ncores`. The function `detectCores` might be useful in this context.
`iseed`	If parallel computation is used, the seed passed on to `clusterSetRNGStream`. Default is `NULL` which means no fixed seed is used.
`eps`	The convergence tolerance for the joint diagonalization.
`maxiter`	The maximum number of iterations for the joint diagonalization.
`...`	Further arguments to be passed to or from methods.

Details

SOBI standardizes `X` with n samples and jointly diagonalizes the autocovariance matrices of the standardized data for a chosen set of lags `tau`, yielding a transformation $\mathbf{W}$ giving the latent variables as $\mathbf{S} = \mathbf{XW}$. Assume, without loss of generality, that the latent components are ordered in decreasing order with respect to the sums of squares of the corresponding "eigenvalues" produced by the joint diagonalization. Under the null hypothesis the final $p-k$ "eigenvalues" of each of the autocovariance matrices equal zero, $\lambda^{\tau}_{p-k} = \cdots = \lambda^{\tau}_{p} = 0$, and their mean square m over all lags can be used as a test statistic in bootstrap-based inference on the true number of latent white noise series.

The function offers four different bootstrapping strategies for generating samples for which the null hypothesis approximately holds, and they are all based on the following general formula:

1. Decompose the SOBI-estimated latent series $\mathbf{S}$ into the postulated signal $\mathbf{S}_1$ and white noise $\mathbf{S}_2$.
2. Take n bootstrap samples $\mathbf{S}_2^*$ of $\mathbf{S}_2$, see the different strategies below.
3. Recombine $\mathbf{S}^* = (\mathbf{S_1}, \mathbf{S_2^*})$ and back-transform $\mathbf{X}^* = \mathbf{S}^* \mathbf{W}^{-1}$.

4. Compute the test statistic based on $\mathbf{X}^*$.

The four different bootstrapping strategies are:

1. `s.boot = "p"`: The first strategy is parametric and simply generates all boostrap samples independently and identically from the standard normal distribution.
2. `s.boot = "np1"`: The second strategy is non-parametric and pools all observed $n(p-k)$ white noise observations together and draws the bootstrap samples from amongst them.
3. `s.boot = "np2"`: The third strategy is non-parametric and proceeds otherwise as the second strategy but acts component-wise. That is, separately for each of the $p - k$ white noise series it pools the observed n white noise observations together and draws the bootstrap samples of that particular latent series from amongst them.
4. `s.boot = "np3"`: The third strategy is non-parametric and instead of drawing the samples univariately as in the second and third strategies, proceeds by resampling n vectors of size $p - k$ from amongst all the observed n white noise vectors.

The function can be run in parallel by setting `ncores` to the desired number of cores (should be less than the number of cores available - 1). When running code in parallel the standard random seed of R is overridden and if a random seed needs to be set it should be passed via the argument `iseed`. The argument `iseed` has no effect in case `ncores` equals 1 (the default value).

This function uses for the joint diagonalization the function `frjd.int`, which does not fail in case of failed convergence but returns the estimate from the final step.

Value

A list of class ictest inheriting from class htest containing:

`statistic`	The value of the test statistic.
`p.value`	The p-value of the test.
`parameter`	The number of bootstrap samples.
`alternative`	Character string specifying the alternative hypothesis.
`k`	The number of latent series that are not white noise used in the testing problem.
`W`	The transformation matrix to the latent series.
`S`	Multivariate time series with the centered source components.
`D`	The underlying eigenvalues of the autocovariance matrix.
`MU`	The location of the data which was subtracted before calculating SOBI.
`tau`	The used set of lags.
`method`	Character string indicating which test was performed.
`data.name`	Character string giving the name of the data.
`s.boot`	Character string denoting which bootstrapping test version was used.

Author(s)

Markus Matilainen, Klaus Nordhausen, Joni Virta

References

Matilainen, M., Nordhausen, K. and Virta, J. (2018), *On the Number of Signals in Multivariate Time Series*. In Deville, Y., Gannot, S., Mason, R., Plumbley, M.D. and Ward, D. (editors) "International Conference on Latent Variable Analysis and Signal Separation", LNCS 10891, 248–258. Springer, Cham., <doi:10.1007/978-3-319-93764-9_24>.

See Also

AMUSE, AMUSEboot, SOBI, tsboot

Examples

```
n <- 1000

A <- matrix(rnorm(16), 4, 4)
s1 <- arima.sim(list(ar = c(0, 0, 0.3, 0.6)), n)
s2 <- arima.sim(list(ma = c(0, 0, -0.3, 0.3)), n)
s3 <- rnorm(n)
s4 <- rnorm(n)

S <- cbind(s1, s2, s3, s4)
X <- S %*% t(A)

boot_res_1 <- SOBIboot(X, k = 1)
boot_res_1

boot_res_2 <- SOBIboot(X, k = 2)
boot_res_2

# Plots of the estimated sources, the last two are white noise
plot(boot_res_2)

# Note that AMUSEboot with lag 1 does not work due to the lack of short range dependencies
AMUSEboot(X, k = 1)

# xts series as input
library(xts)
data(sample_matrix)
X2 <- as.xts(sample_matrix)
boot_res_xts <- SOBIboot(X2, k = 2)
plot(boot_res_xts, multi.panel = TRUE)

# zoo series as input
X3 <- as.zoo(X)
boot_res_zoo <- SOBIboot(X3, k = 2)
plot(boot_res_zoo)
```

SOBIladle *Ladle Estimator to Estimate the Number of White Noise Components in SOS with SOBI*

Description

The ladle estimator uses the joint diagonalization "eigenvalues" and "eigenvectors" of several autocovariance matrices to estimate the number of white noise components in second-order source separation (SOS).

Usage

```
SOBIladle(X, ...)

## Default S3 method:
SOBIladle(X, tau = 1:12, l = 20, sim = c("geom", "fixed"), n.boot = 200,
ncomp = ifelse(ncol(X) > 10, floor(ncol(X)/log(ncol(X))), ncol(X) - 1),
maxiter = 1000, eps = 1e-06, ...)
## S3 method for class 'ts'
SOBIladle(X, ...)
## S3 method for class 'xts'
SOBIladle(X, ...)
## S3 method for class 'zoo'
SOBIladle(X, ...)
```

Arguments

`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. Missing values are not allowed.
`tau`	The lags for the SOBI autocovariance matrices.
`l`	If `sim = "geom"` then `l` is the success probability of the geometric distribution from where the bootstrap block lengths for the stationary bootstrap are drawn. If `sim = "fixed"` then `l` is the fixed block length for the fixed block bootstrap.
`sim`	If `"geom"` (default) then the stationary bootstrap is used. If `"fixed"` then the fixed block bootstrap is used.
`n.boot`	The number of bootstrapping samples. See `tsboot` for details.
`ncomp`	The number of components among which the ladle estimator is to be searched. Must be between `0` and `ncol(X)-1`. The default follows the recommendation of Luo and Li (2016).
`maxiter`	Maximum number of iterations.
`eps`	Convergence tolerance.
`...`	Further arguments to be passed to or from methods.

Details

SOBI standardizes X with n samples and jointly diagonalizes the autocovariance matrices of the standardized data for a chosen set of lags tau, yielding a transformation $\mathbf{W}$ giving the latent variables as $\mathbf{S} = \mathbf{XW}$. Assume, without loss of generality, that the latent components are ordered in decreasing order with respect to the sums of squares of the corresponding "eigenvalues" produced by the joint diagonalization. Under the assumption that we have k non-white-noise components, the final $p-k$ "eigenvalues" of each of the autocovariance matrices equal zero, $\lambda^\tau_{p-k} = \cdots = \lambda^\tau_p = 0$.

The change point from non-zero eigenvalues to zero eigenvalues is visible in the joint diagonalization "eigenvectors" of the autocovariance matrices as an increase in their boostrap variablity. Similarly, before the change point, the squared eigenvalues decrease in magnitude and afterwards they stay constant. The ladle estimate combines the scaled eigenvector bootstrap variability with the scaled eigenvalues to estimate the number of non-white-noise components. The estimate is the value of $k = 0, \ldots,$ ncomp where the combined measure achieves its minimum value.

This function uses for the joint diagonalization the function frjd.int, which does not fail in case of failed convergence but returns the estimate from the final step.

Value

A list of class ladle containing:

method	The string SOBI.
k	The estimated number of non-white-noise components.
fn	The vector giving the measures of variation of the eigenvectors using the bootstrapped eigenvectors for the different number of components.
phin	Normalized sums of squared eigenvalues of the SOBI matrices.
data.name	The name of the data for which the ladle estimate was computed.
gn	The main criterion for the ladle estimate - the sum of fn and phin. k is the value where gn takes its minimum.
lambda	The sums of squared eigenvalues of the SOBI matrices.
W	The transformation matrix to the source components. Also known as the unmixing matrix.
S	Multivariate time series with the centered source components.
MU	The location of the data which was subtracted before calculating the source components.
sim	The used boostrapping technique, either "geom" or "fixed".
tau	The used set of lags for the SOBI autocovariance matrices.

Author(s)

Klaus Nordhausen, Joni Virta

References

Nordhausen, K. and Virta, J.(2018), *Ladle Estimator for Time Series Signal Dimension*. In 2018 IEEE Statistical Signal Processing Workshop (SSP), pp. 428–432, <doi:10.1109/SSP.2018.8450695>.

Luo, W. and Li, B. (2016), *Combining Eigenvalues and Variation of Eigenvectors for Order Determination*, Biometrika, 103. 875–887. <doi:10.1093/biomet/asw051>

See Also

AMUSE, SOBI, AMUSEladle, frjd.int

Examples

```
n <- 1000

s1 <- arima.sim(n = n, list(ar = 0.6, ma = c(0, -0.4)))
s2 <- arima.sim(n = n, list(ar = c(0, 0.1,0.3), ma = c(0.2, 0.4)))
s3 <- arima.sim(n = n, list(ar = c(0, 0.8)))
Snoise <- matrix(rnorm(5*n), ncol = 5)
S <- cbind(s1, s2, s3, Snoise)

A <- matrix(rnorm(64), 8, 8)
X <- S %*% t(A)

ladle_SOBI <- SOBIladle(X, l = 20, sim = "geom")

# The estimated number of non-white-noise components
summary(ladle_SOBI)

# The ladle plot
ladleplot(ladle_SOBI)

# Time series plots of the estimated components
plot(ladle_SOBI)

# Note that AMUSEladle with lag 1 does not work due to the lack of short range dependencies
ladle_AMUSE <- AMUSEladle(X)

summary(ladle_AMUSE)
ladleplot(ladle_AMUSE)

# xts series as input
library(xts)
data(sample_matrix)
X2 <- as.xts(sample_matrix)
ladle_SOBI_xts <- SOBIladle(X2, l = 20, sim = "geom")
plot(ladle_SOBI_xts, multi.panel = TRUE)

# zoo series as input
X3 <- as.zoo(X)
ladle_SOBI_zoo <- SOBIladle(X3, l = 20, sim = "geom")
plot(ladle_SOBI_zoo)
```

---

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 = c("rectangle", "alllag", "alldir", "big"), thres = 0.8, ...)

## S3 method for class 'summary.tssdr'
print(x, digits = 3, ...)
## S3 method for class 'summary.tssdr'
components(x, ...)
## S3 method for class 'summary.tssdr'
coef(object, ...)
## S3 method for class 'summary.tssdr'
plot(x, main = "The response and the chosen directions", ...)
```

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.
	In methods for class 'summary.tssdr' only:
`x`	An object of class summary.tssdr
`digits`	The number of digits when printing an object of class summary.tssdr. Default is 3
`main`	A title for a plot when printing an object of class summary.tssdr.

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 π (`thres`). Let λ_{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 $(i_1, j_1), \ldots, (i_r, j_r)$ 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 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. and 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 a hybrid of TSIR and TSAVE is TSSH (Time series SIR SAVE Hybrid). For summary of an object of class tssdr, see `summary.tssdr`.

Usage

```
tssdr(y, X, ...)

## Default S3 method:
tssdr(y, X, algorithm = c("TSIR", "TSAVE", "TSSH"), k = 1:12, H = 10, weight = 0.5,
      method = c("frjd", "rjd"), eps = 1e-06, maxiter = 1000, ...)
## S3 method for class 'ts'
tssdr(y, X, ...)
## S3 method for class 'xts'
tssdr(y, X, ...)
## S3 method for class 'zoo'
tssdr(y, X, ...)

## S3 method for class 'tssdr'
print(x, digits = 3, ...)
## S3 method for class 'tssdr'
components(x, ...)
## S3 method for class 'tssdr'
plot(x, main = "The response and the directions", ...)
```

Arguments

`y`	A numeric vector or a time series object of class `ts`, `xts` or `zoo` (same type as X). Missing values are not allowed.
`X`	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo` (same type as y). 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 "rjd" and "frjd". Default is "frjd".
eps	Convergence tolerance.
maxiter	The maximum number of iterations.
...	Further arguments to be passed to or from methods.
	In methods for class 'tssdr' only:
x	An object of class tssdr
digits	The number of digits when printing an object of class tssdr. Default is 3
main	A title for a plot when printing an object of class tssdr.

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 $\widehat{\mathbf{A}}_{\mathbf{jh}} = \frac{\mathbf{1}}{\mathbf{T_{jh}}}\sum_{\mathbf{t=1}}^{\mathbf{T_{jh}}}(\mathbf{Z_{jh}})_{\mathbf{t}}$ and $\widehat{\mathbf{A}}_j = (\widehat{\mathbf{A}}_{j1}, \ldots, \widehat{\mathbf{A}}_{jH})'$. Then for algorithm TSIR matrix

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

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

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

For TSSH then matrix

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

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

The algorithms find an orthogonal matrix $\mathbf{U} = (\mathbf{u_1}, \ldots, \mathbf{u_p})'$ by maximizing, for $b = 0, 1$ or 2,

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

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 λ_{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 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. and Oja H. (2017), *Supervised Dimension Reduction for Multivariate Time Series*, Econometrics and Statistics, 4, 57–69.

Matilainen M., Croux C., Nordhausen K. and Oja H. (2019), *Sliced Average Variance Estimation for Multivariate Time Series*. Statistics: A Journal of Theoretical and Applied Statistics, 53, 630–655, <doi:10.1080/02331888.2019.1605515>.

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

Cook, R. and 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))
```

---

vSOBI *A Variant of SOBI for Blind Source Separation*

Description

The vSOBI (variant of Second Order Blind Identification) method for the blind source separation of 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 = c("pow", "lcosh"),
                  ordered = FALSE, acfk = NULL, original = TRUE, alpha = 0.05, ...)
## S3 method for class 'ts'
vSOBI(X, ...)
## S3 method for class 'xts'
vSOBI(X, ...)
## S3 method for class 'zoo'
vSOBI(X, ...)
```

Arguments

X	A numeric matrix or a multivariate time series object of class `ts`, `xts` or `zoo`. 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 `"pow"` (default) and `"lcosh"`.
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.
alpha	Alpha level for linear correlation detection. Default is 0.05.
...	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 `diag(p)` as an initial value for an orthogonal matrix $\mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_p)'$.

Matrix $\mathbf{T}_{ik}$ is a partial derivative of $\mathbf{D}_k(\mathbf{U})$ with respect to $\mathbf{u}_i$. Then $\mathbf{T}_k = (\mathbf{T}_{1k}, \ldots, \mathbf{T}_{pk})'$, 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' containing the following components:

W	The estimated unmixing matrix. If ordered = TRUE, the rows are ordered according to the order of the components.
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 original = FALSE, the sources with linear autocorrelation are replaced by their ARMA residuals.
MU	The means of the original series.

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

`Sraw`	The ordered original estimated sources as time series object standardized to have mean 0 and unit variances. Returned only if `original = FALSE`.
`fits`	The ARMA fits for the components with linear autocorrelation.
`armaeff`	A logical vector. Is TRUE 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

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 *Logarithmic Returns of Exchange Rates of 7 Currencies Against US Dollar*

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 $\mathbf{r}_{AUD,t}$ of the exchange rates of AUD against US Dollar.

`CAD` The weekly logarithmic returns $\mathbf{r}_{CAD,t}$ of the exchange rates of CAD against US Dollar.

`NOK` The weekly logarithmic returns $\mathbf{r}_{NOK,t}$ of the exchange rates of NOK against US Dollar.

`SGD` The weekly logarithmic returns $\mathbf{r}_{SGD,t}$ of the exchange rates of SGD against US Dollar.

`SEK` The weekly logarithmic returns $\mathbf{r}_{SEK,t}$ of the exchange rates of SEK against US Dollar.

`CHF` The weekly logarithmic returns $\mathbf{r}_{CHF,t}$ of the exchange rates of CHF against US Dollar.

`GBP` The weekly logarithmic returns $\mathbf{r}_{GBP,t}$ of the exchange rates of GBP against US Dollar.

Details

The daily exhange 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 $\mathbf{x}_{j,t}$ be the exchange rates of j against US Dollar. Then

$$\mathbf{r}_{j,t} = \log(\mathbf{x}_{j,t}) - \log(\mathbf{x}_{j,t-1}),$$

where $t = 1, \ldots, 605$ and $j = AUD, CAD, NOK, SGD, SEK, CHF, GBP$. The six missing values in $\mathbf{r}_{SEK,t}$ 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. (2018). 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. (2019), *Extracting Conditionally Heteroskedastic Components Using Independent Component Analysis*. Accepted for publication in Journal of Time Series Analysis. Available at http://arxiv.org/abs/1811.10963v1.

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