

Package ‘JADE’

February 14, 2012

Type Package

Title JADE and other BSS methods as well as some BSS performance criteria

Version 1.1-0

Date 2012-02-14

Author Klaus Nordhausen, Jean-Francois Cardoso, Jari Miettinen, Hannu Oja, Esa Ollila, Sara Taskinen

Maintainer Klaus Nordhausen <klaus.nordhausen@uta.fi>

Depends clue

Suggests ICS, ICSNP

Description The package ports JF Cardoso’s JADE algorithm as well as his function for joint diagonalization. There are also several other blind source separation (BSS) methods like AMUSE and SOBI as well as some criteria for performance evaluation of BSS algorithms.

License GPL (>= 2)

Repository CRAN

Date/Publication 2012-02-14 13:09:54

R topics documented:

JADE-package	2
amari.error	2
AMUSE	4
bss.components	5
coef.bss	6
ComonGAP	7
djd	8
JADE	9
MD	11

multscatter	12
plot.bss	13
print.bss	14
rjd	15
SIR	16
SOBI	17

Index	19
--------------	-----------

JADE-package	<i>JADE and other BSS methods as well as some BSS performance criteria</i>
--------------	--

Description

The package ports JF Cardoso's JADE algorithm as well as his function for joint diagonalization. There are also several other blind source separation (BSS) methods like AMUSE and SOBI as well as some criteria for performance evaluation of BSS algorithms.

Details

Package: JADE
 Type: Package
 Version: 1.1-0
 Date: 2012-02-14
 License: GPL (>= 2)

Author(s)

Klaus Nordhausen, Jean-Francois Cardoso, Jari Miettinen, Hannu Oja, Esa Ollila, Sara Taskinen
 Maintainer: Klaus Nordhausen <klaus.nordhausen@uta.fi>

amari.error	<i>Amari Error</i>
-------------	--------------------

Description

Computes the Amari Error to evaluate the performance of an ICA algorithm.

Usage

```
amari.error(W.hat, A, standardize = F)
```

Arguments

W.hat	The estimated square unmixing matrix W .
A	The true square mixing matrix A .
standardize	Logical value if A and W .hat need to be standardized. Default is FALSE.

Details

The Amari Error can be used in simulation studies to evaluate the performance of an ICA algorithm. The Amari error is permutation invariant but not scale invariant. Therefore if different algorithms should be compared the matrices should be scaled in the same way. If `standardize` is TRUE, this will be done by the function by standardizing 'W.hat' and the inverse of 'A' in such a way, that every row has length 1, the largest absolute value of the row has a positive sign and the rows are ordered decreasingly according to their largest values.

Note that this function assumes the ICA model is $X = SA'$, as is assumed by [JADE](#) and [ics](#). However [fastICA](#) and [PearsonICA](#) assume $X = SA$. Therefore matrices from those functions have to be transposed first.

The Amari Error is scaled in such a way, that it takes a value between 0 and 1. And 0 corresponds to an optimal separation.

Value

The value of the Amari Error.

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

References

Amari, S., Cichocki, A. and Yang, H.H. (1996), A new learning algorithm for blind signal separation, Advances in Neural Information Processing Systems, 8, 757–763.

See Also

[ComonGAP](#), [SIR](#)

Examples

```
S <- cbind(rt(1000, 4), rnorm(1000), runif(1000))
A <- matrix(rnorm(9), ncol = 3)
X <- S %*% t(A)

W.hat <- JADE(X, 3)$W
amari.error(W.hat, A)
amari.error(W.hat, A, TRUE)
```

Description

AMUSE method for the second order blind source separation problem. The function estimates the unmixing matrix in a second order stationary source separation model by jointly diagonalizing the covariance matrix and an autocovariance matrix at lag k .

Usage

```
AMUSE(x, ...)  
  
## Default S3 method:  
AMUSE(x, k = 1, ...)  
## S3 method for class 'ts'  
AMUSE(x, ...)
```

Arguments

<code>x</code>	a numeric matrix or a multivariate time series object of class <code>ts</code> . Missing values are not allowed.
<code>k</code>	integer lag for the autocovariance matrix, must be larger than 0. Default is 1.
<code>...</code>	further arguments to be passed to or from methods.

Details

The chosen lag k has a huge effect on the performance. It should be chosen such that the eigenvalues of autocovariance matrix are distinct. The function assumes always as many sources as there are time series.

Value

A list with class `'bss'` containing the following components:

<code>W</code>	estimated unmixing matrix.
<code>EV</code>	eigenvectors of autocovariance matrix.
<code>k</code>	lag of the autocovariance matrix used.
<code>S</code>	estimated sources as time series object standardized to have mean 0 and unit variances.

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

References

Tong, L., Soon, V.C., Huang, Y.F. and Liu, R. (1990), *AMUSE: a new blind identification algorithm*, in *Proceedings of IEEE International Symposium on Circuits and Systems 1990*, 1784–1787.

Miettinen, J., Nordhausen, K., Oja, H. and Taskinen, S. (2012), *Statistical properties of a blind source separation estimator for stationary time series*, submitted, ??–??.

See Also

[ts](#)

Examples

```
# creating some toy data
A<- matrix(rnorm(9),3,3)
s1 <- arima.sim(list(ar=c(0.3,0.6)),1000)
s2 <- arima.sim(list(ma=c(-0.3,0.3)),1000)
s3 <- arima.sim(list(ar=c(-0.8,0.1)),1000)

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

res1<-AMUSE(X)
res1
coef(res1)
plot(res1) # compare to plot.ts(S)
MD(coef(res1),A)

# input of a time series
X2<- ts(X, start=c(1961, 1), frequency=12)
plot(X2)
res2<-AMUSE(X2, k=2)
plot(res2)
```

bss.components

Function to Extract Estimated Sources from an Object of Class bss

Description

Extracts the sources estimated by an bss method.

Usage

```
bss.components(object)
```

Arguments

object object of class bss

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

Examples

```
A<- matrix(rnorm(9),3,3)
s1 <- arima.sim(list(ar=c(0.3,0.6)),1000)
s2 <- arima.sim(list(ma=c(-0.3,0.3)),1000)
s3 <- arima.sim(list(ar=c(-0.8,0.1)),1000)

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

res1<-AMUSE(X)
head(bss.components(res1))
colMeans(bss.components(res1))
cov(bss.components(res1))
```

coef.bss

Coefficients of a bss Object

Description

Extracts the estimated unmixing matrix from an object of class bss.

Usage

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

Arguments

object object of class bss.
 ... further arguments to be passed to or from methods.

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

Examples

```
A<- matrix(rnorm(9),3,3)
s1 <- arima.sim(list(ar=c(0.3,0.6)),1000)
s2 <- arima.sim(list(ma=c(-0.3,0.3)),1000)
s3 <- arima.sim(list(ar=c(-0.8,0.1)),1000)

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

```
res1<-AMUSE(X)
coef(res1)
coef(res1) %*% A # should be a matrix with one dominant element in each row and column
```

ComonGAP

Comon's Gap

Description

Comon's GAP criterion to evaluate the performance of an ICA algorithm.

Usage

```
ComonGAP(A, A.hat)
```

Arguments

A	The true square mixing matrix.
A.hat	The estimated square mixing matrix.

Details

Comon's GAP criterion is permutation and scale invariant. It can take every positive value and 0 corresponds to an optimal separation. If A is however nearly singular the values of the criterion can be huge.

Note that this function assumes the ICA model is $X = SA'$, as is assumed by [JADE](#) and [ics](#). However [fastICA](#) and [PearsonICA](#) assume $X = SA$. Therefore matrices from those functions have to be transposed first.

Value

The value of the Comon's GAP.

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

References

Comon, P., (1994), *Independent Component Analysis, A new concept?*, *Signal Processing*, **36**, 287–314.

See Also

[amari.error](#), [SIR](#)

Examples

```
S <- cbind(rt(1000, 4), rnorm(1000), runif(1000))
A <- matrix(rnorm(9), ncol = 3)
X <- S %*% t(A)

A.hat <- JADE(X, 3)$A
ComonGAP(A, A.hat)
```

djd

Function for Joint Diagonalization of k Square Matrices in a Deflation Based Manner

Description

This function jointly diagonalizes k real-valued square matrices by searching an orthogonal matrix in a deflation based manner.

Usage

```
djd(X, G = "pow", r = 2, eps = 1e-06, maxiter = 100)
```

Arguments

X	an array containing the k p times p real valued matrices of dimension $c(p, p, k)$.
G	criterion function used for the the algorithm. Options are pow and log. See details.
r	power value used if $G = \text{"pow"}$. 0 is not meaningful for this value. See details.
eps	convergence tolerance.
maxiter	maximum number of iterations.

Details

Denote the square matrices as $A_i, i = 1, \dots, k$. This algorithm searches then an orthogonal matrix W so that $D_i = W' A_i W$ is diagonal for all i . If the A_i commute then there is an exact solution. If not, the function will perform an approximate joint diagonalization by maximizing $\sum G(w_j' A_i w_j)$ where w_j are the orthogonal vectors in W . The directions are found in a deflation based manner where the initial values are the eigenvectors of A_1 which plays hence a special role.

The function G can be choosen to be of the form $G(x) = x^r$ or $G(x) = \log(x)$.

Value

The matrix W

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

Examples

```

Z <- matrix(runif(9), ncol = 3)
U <- eigen(Z %*% t(Z))$vectors
D1 <- diag(runif(3))
D2 <- diag(runif(3))
D3 <- diag(runif(3))
D4 <- diag(runif(3))

X.matrix <- array(0, dim=c(3, 3, 4))
X.matrix[, ,1] <- t(U) %*% D1 %*% U
X.matrix[, ,2] <- t(U) %*% D2 %*% U
X.matrix[, ,3] <- t(U) %*% D3 %*% U
X.matrix[, ,4] <- t(U) %*% D4 %*% U

W1 <- djd(X.matrix)
round(U %*% W1, 4) # should be a signed permutation
                  # matrix if W1 is correct.

W2 <- djd(X.matrix, r=1)
round(U %*% W2, 4) # should be a signed permutation
                  # matrix if W2 is correct.

W3 <- djd(X.matrix, G="1")
round(U %*% W3, 4) # should be a signed permutation
                  # matrix if W3 is correct.

```

 JADE

JADE algorithm for ICA

Description

This is an **R** version of Cardoso's JADE ICA algorithm for real data ported from matlab. The ported version is 1.5, some minor changes compared to the matlab function are explained in the details section. The matlab code can be found for example on the ICA central homepage.

Usage

```
JADE(X, n.comp = NULL, eps = 1e-06, maxiter = 100, na.action = na.fail)
```

Arguments

X	Numeric data matrix or dataframe.
n.comp	Number of components to extract.
eps	Convergence tolerance.
maxiter	Maximum number of iterations.
na.action	A function which indicates what should happen when the data contain 'NA's. Default is to fail.

Details

Some minor modifications were done when porting the function to **R**, and they are:

- 1 the model assumed here is $X = SA' + \mu$. Therefore S and X have one row per observation. Note that this still differs from the model definition in **R** of FastICA and PearsonICA but agrees with `ics`.
- 2 The whitening covariance matrix is divided by $n-1$ and not n (n = number of observations).
- 3 The initial value for the joint diagonalisation is always I .
- 4 The original `eps` would be $\frac{1}{100\sqrt{n}}$.

Worth mentioning is also that the estimated independent components S are scaled to unit variance and are ordered in such a way, that the most energetic component comes first. The signs of the unmixing matrix W are fixed such, that the first column of W has positive elements.

For further details see also the documentation of the original matlab code ("MatlabjadeR.m") on the ICA central homepage (<http://www.tsi.enst.fr/icacentral/>).

Value

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

A	The estimated mixing matrix.
W	The estimated unmixing matrix.
S	Dataframe with the estimated independent components.
Xmu	The location of the original data.

Author(s)

Jean-Francois Cardoso. Ported to **R** by Klaus Nordhausen, <klaus.nordhausen@uta.fi>

References

Cardoso, J.-F. and Souloumiac, A., (1993), *Blind beamforming for non Gaussian signals*, IEE Proceedings-F, **140**, 362–370. <ftp://sig.enst.fr/pub/jfc/Papers/iee.ps.gz>.

Examples

```
# 3 source and 3 signals

S <- cbind(rt(1000, 4), rnorm(1000), runif(1000))
A <- matrix(rnorm(9), ncol = 3)
X <- S %*% t(A)
res<-JADE(X,3)
res$A
res$W
res$S[1:10,]
(sweep(X,2,res$Xmu) %*% t(res$W))[1:10,]
round(res$W %*% A,4)

# 2 sources and 3 signals
```

```

S2 <- cbind(rt(1000, 4), rnorm(1000))
A2 <- matrix(rnorm(6), ncol = 2)
X2 <- S2 %*% t(A2)
res2 <- JADE(X2, 2)
res2$A
res2$W
res2$S[1:10,]
(sweep(X2, 2, res2$Xmu) %*% t(res2$W))[1:10,]
SIR(S2, res2$S)

```

MD

Minimum Distance index MD

Description

Computes the Minimum Distance index MD to evaluate the performance of an ICA algorithm.

Usage

```
MD(W.hat, A)
```

Arguments

W.hat	The estimated square unmixing matrix W.
A	The true square mixing matrix A.

Details

$$MD(\hat{W}, A) = \frac{1}{\sqrt{p-1}} \inf_{PD} \|PD\hat{W}A - I\|,$$

where P is a permutation matrix and D a diagonal matrix with nonzero diagonal entries.

The step that minimizes the index of the set over all permutation matrix can be expressed as a linear sum assignment problem (LSAP) for which we use as solver the Hungarian method implemented as `solve_LASP` in the **clue** package.

Note that this function assumes the ICA model is $X = SA'$, as is assumed by [JADE](#) and [ics](#). However [fastICA](#) and [PearsonICA](#) assume $X = SA$. Therefore matrices from those functions have to be transposed first.

The MD index is scaled in such a way, that it takes a value between 0 and 1. And 0 corresponds to an optimal separation.

Value

The value of the MD index.

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

References

Ilmonen, P., Nordhausen, K., Oja, H. and Ollila, E. (2010): A New Performance Index for ICA: Properties, Computation and Asymptotic Analysis. In Vigneron, V., Zarzoso, V., Moreau, E., Gribonval, R. and Vincent, E. (editors) Latent Variable Analysis and Signal Separation, 229–236, Springer.

See Also

[ComonGAP](#), [SIR](#), [amari.error](#), [solve_LASP](#)

Examples

```
S <- cbind(rt(1000, 4), rnorm(1000), runif(1000))
A <- matrix(rnorm(9), ncol = 3)
X <- S %*% t(A)

W.hat <- JADE(X, 3)$W
MD(W.hat, A)
```

multscatter

Function to Compute Several Scatter Matrices for the Same Data

Description

The function can be used to compute several scatter matrices for the same data.

Usage

```
multscatter(scatterlist, X, toshape = TRUE)
```

Arguments

scatterlist	a vector with the names of the scatter matrices to be computed. Note that each of these functions should only return a matrix of size p times p .
X	the n times p data matrix for which the scatter should be computed.
toshape	logical, whether scatter matrices should be converted to shape matrices. If TRUE, all matrices will have determinant 1.

Details

It is important that the functions do not need any additional input and that they return only the p times p scatter matrix. Hence it might be sometimes necessary to write wrappers for some of the functions. See examples.

Value

An array of dimension $c(p,p,k)$ where k is the number of scatter matrices.

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

Examples

```
# example requires the packages ICS and ICSNP
library(ICSNP)
X <- cbind(rexp(1000), rt(1000,6), runif(1000))

my.tM1 <- function(X,df=1) tM(X,)$V
my.tM2 <- function(X,df=2) tM(X,)$V

multscatter(c("cov", "cov4", "HP1.shape", "my.tM1", "my.tM2"), X)
multscatter(c("cov", "cov4", "HP1.shape", "my.tM1", "my.tM2"), X, toshape=FALSE)
```

plot.bss

Plotting an Object of Class bss

Description

Plots the estimated sources resulting from an bss method. If the bss method is based on second order assumptions and returned the sources as a time series object it will plot the sources using `plot.ts`, otherwise it will plot a scatter plot matrix using `pairs`.

Usage

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

Arguments

`x` object of class `bss`.
`...` further arguments to be passed to or from methods.

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

See Also

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

Examples

```
A<- matrix(rnorm(9),3,3)
s1 <- arima.sim(list(ar=c(0.3,0.6)),1000)
s2 <- arima.sim(list(ma=c(-0.3,0.3)),1000)
s3 <- arima.sim(list(ar=c(-0.8,0.1)),1000)

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

res1 <- AMUSE(X)
plot(res1)
# not so useful:
plot(res1, plot.type = "single", col=1:3)

# not meaningful for this data
res2 <- JADE(X)
plot(res2)
```

print.bss

Printing an Object of Class bss

Description

Prints an object of class bss. It prints all elements of the list of class bss except the component S which is the source matrix.

Usage

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

Arguments

x object of class bss.
... further arguments to be passed to or from methods.

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

rjd

*Joint Diagonalization of Real Matrices***Description**

This is an **R** version of Cardoso's rjd matlab function for joint diagonalization of k real-valued square matrices.

Usage

```
rjd(X, eps = 1e-06, maxiter = 100, na.action = na.fail)
```

Arguments

X	A matrix of k stacked $p \times p$ matrices with dimension $c(kp, p)$ or an array with dimension $c(p, p, k)$.
eps	Convergence tolerance.
maxiter	Maximum number of iterations.
na.action	A function which indicates what should happen when the data contain 'NA's. Default is to fail.

Details

Denote the square matrices as $A_i, i = 1, \dots, k$. This algorithm searches then an orthogonal matrix V so that $D_i = V' A_i V$ is diagonal for all i . If the A_i commute then there is an exact solution. If not, the function will perform an approximate joint diagonalization by trying to make the D_i as diagonal as possible.

Cardoso points out that notion of approximate joint diagonalization is ad hoc and very small values of eps make in that case not much sense since the diagonality criterion is ad hoc itself.

Value

A list with the components

V	An orthogonal matrix.
D	A stacked matrix with the diagonal matrices or an array with the diagonal matrices. The form of the output depends on the form of the input.

Author(s)

Jean-Francois Cardoso. Ported to **R** by Klaus Nordhausen, <klaus.nordhausen@uta.fi>

References

Cardoso, J.-F. and Souloumiac, A., (1996), *Jacobi angles for simultaneous diagonalization*, SIAM J. Mat. Anal. Appl., **17**, 161–164.

Examples

```

Z <- matrix(runif(9), ncol = 3)
U <- eigen(Z %% t(Z))$vectors
D1 <- diag(runif(3))
D2 <- diag(runif(3))
D3 <- diag(runif(3))
D4 <- diag(runif(3))

X.matrix <- rbind(t(U) %% D1 %% U, t(U) %% D2 %% U,
                 t(U) %% D3 %% U, t(U) %% D4 %% U)
res.matrix <- rjd(X.matrix)
res.matrix$V
round(U %% res.matrix$V, 4) # should be a signed permutation
                           # matrix if V is correct.

round(res.matrix$D, 4)

X.array <- aperm(array(t(X.matrix), dim = c(3,3,4)), c(2,1,3))

res.array <- rjd(X.array)
round(res.array$D, 4)

```

SIR

*Signal to Interference Ratio***Description**

Computes the signal to interference ratio between true and estimated signals

Usage

```
SIR(S, S.hat)
```

Arguments

S	Matrix or dataframe with the true numeric signals.
S.hat	Matrix or dataframe with the estimated numeric signals.

Details

The signal to interference ratio is measured in dB and values over 20 are thought to be good. It is scale and permutation invariant and can be seen as measuring the correlation between the matched true and estimated signals.

Value

The value of the signal to interference ratio.

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

References

Eriksson, J., Karvanen, J. and Koivunen, V. (2000), Source distribution adaptive maximum likelihood estimation in ICA model, Proceedings of the second international workshop on independent component analysis and blind source separation (ICA 2000), 227–232.

See Also

[amari.error](#), [ComonGAP](#)

Examples

```
S <- cbind(rt(1000, 4), rnorm(1000), runif(1000))
A <- matrix(rnorm(9), ncol = 3)
X <- S %*% t(A)

S.hat <- JADE(X, 3)$S
SIR(S, S.hat)
```

SOBI

SOBI Method for Blind Source Separation

Description

The SOBI method for the second order blind source separation problem. The function estimates the unmixing matrix in a second order stationary source separation model by jointly diagonalizing the covariance matrix and several autocovariance matrices at different lags.

Usage

```
SOBI(X, ...)
```

Default S3 method:
SOBI(X, k=12, method="rjd", eps = 1e-06, maxiter = 100, ...)
S3 method for class 'ts'
SOBI(X, ...)

Arguments

X	a numeric matrix or a multivariate time series object of class <code>ts</code> . Missing values are not allowed.
k	if a single integer, then the lags 1:k are used, if an integer vector, then these are used as the lags.
method	method to use for the joint diagonalization, options are djd and rjd .

eps maximum number of iterations.
 maxiter convergence tolerance.
 ... further arguments to be passed to or from methods.

Value

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

W estimated unmixing matrix.
 k lags used.
 method method used for the joint diagonalization.
 S estimated sources as time series objected standardized to have mean 0 and unit variances.

Author(s)

Klaus Nordhausen, <klaus.nordhausen@uta.fi>

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.

See Also

[ts](#)

Examples

```
# creating some toy data
A<- matrix(rnorm(9),3,3)
s1 <- arima.sim(list(ar=c(0.3,0.6)),1000)
s2 <- arima.sim(list(ma=c(-0.3,0.3)),1000)
s3 <- arima.sim(list(ar=c(-0.8,0.1)),1000)

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

res1<-SOBI(X)
res1
coef(res1)
plot(res1) # compare to plot.ts(S)
MD(coef(res1),A)

# input of a time series
X2<- ts(X, start=c(1961, 1), frequency=12)
plot(X2)
res2<-SOBI(X2, k=c(5,10,1,4,2,9,10))
plot(res2)
```

Index

- *Topic **array**
 - djd, [8](#)
 - rjd, [15](#)
- *Topic **methods**
 - coef.bss, [6](#)
 - plot.bss, [13](#)
 - print.bss, [14](#)
- *Topic **multivariate**
 - amari.error, [2](#)
 - AMUSE, [4](#)
 - bss.components, [5](#)
 - ComonGAP, [7](#)
 - JADE, [9](#)
 - MD, [11](#)
 - multscatter, [12](#)
 - SIR, [16](#)
 - SOBI, [17](#)
- *Topic **package**
 - JADE-package, [2](#)
- *Topic **ts**
 - AMUSE, [4](#)
 - SOBI, [17](#)

amari.error, [2](#), [7](#), [12](#), [17](#)

AMUSE, [4](#)

bss.components, [5](#)

coef.bss, [6](#)

ComonGAP, [3](#), [7](#), [12](#), [17](#)

djd, [8](#), [17](#)

JADE, [3](#), [7](#), [9](#), [11](#)

JADE-package, [2](#)

MD, [11](#)

multscatter, [12](#)

pairs, [13](#)

plot.bss, [13](#)

plot.ts, [13](#)

print.bss, [14](#)

rjd, [15](#), [17](#)

SIR, [3](#), [7](#), [12](#), [16](#)

SOBI, [17](#)

ts, [4](#), [5](#), [17](#), [18](#)