Journal of Statistical Software

January 2017, Volume 76, Issue 2. doi: 10.18637/jss.v076.i02

# Blind Source Separation Based on Joint Diagonalization in R: The Packages JADE and BSSasymp

**Jari Miettinen**
University of Jyvaskyla

**Klaus Nordhausen**
University of Turku

**Sara Taskinen**
University of Jyvaskyla

**Abstract**

Blind source separation (BSS) is a well-known signal processing tool which is used to solve practical data analysis problems in various fields of science. In BSS, we assume that the observed data consists of linear mixtures of latent variables. The mixing system and the distributions of the latent variables are unknown. The aim is to find an estimate of an unmixing matrix which then transforms the observed data back to latent sources. In this paper we present the R packages **JADE** and **BSSasymp**. The package **JADE** offers several BSS methods which are based on joint diagonalization. Package **BSSasymp** contains functions for computing the asymptotic covariance matrices as well as their data-based estimates for most of the BSS estimators included in package **JADE**. Several simulated and real datasets are used to illustrate the functions in these two packages.

*Keywords*: independent component analysis, multivariate time series, nonstationary source separation, performance indices, second order source separation.

## 1. Introduction

The blind source separation (BSS) problem is, in its most simple form, the following: Assume that observations $x_1, \ldots, x_n$ are $p$-variate vectors whose components are linear combinations of the components of $p$-variate unobservable zero mean vectors $z_1, \ldots, z_n$. If we consider $p$-variate vectors $x$ and $z$ as row vectors (to be consistent with the programming language R, R Core Team 2016), the BSS model can be written as

$$x = zA^\top + \mu, \tag{1}$$

where $A$ is an unknown full rank $p \times p$ mixing matrix and $\mu$ is a $p$-variate location vector. The goal is then to estimate an unmixing matrix, $W = A^{-1}$, based on the $n \times p$ data matrix

$X = [x_1^\top, \ldots, x_n^\top]^\top$, such that $z_i = (x_i - \mu)W^\top$, $i = 1, \ldots, n$. Notice that BSS can also be applied in cases where the dimension of $x$ is larger than that of $z$ by applying a dimension reduction method as a first stage. In this paper we, however, restrict to the case where $A$ is a square matrix.

The unmixing matrix $W$ cannot be estimated without further assumptions on the model. There are three major BSS models which differ in their assumptions made upon $z$: In independent component analysis (ICA), which is the most popular BSS approach, it is assumed that the components of $z$ are mutually independent and at most one of them is Gaussian. ICA applies best to cases where also $z_1, \ldots, z_n$ are independent and identically distributed (iid). The two other main BSS models, the second order source separation (SOS) model and the second order nonstationary source separation (NSS) model, utilize temporal or spatial dependence within each component. In the SOS model, the components are assumed to be uncorrelated weakly (second-order) stationary time series with different time dependence structures. The NSS model differs from the SOS model in that the variances of the time series components are allowed to be nonstationary. All these three models will be defined in detail later in this paper.

None of the three models has a unique solution. This can be seen by choosing any $p \times p$ matrix $C$ from the set

$$\mathcal{C} = \{C : \text{ each row and column of } C \text{ has exactly one non-zero element}\}. \tag{2}$$

Then $C$ is invertible, $A^* = AC^{-1}$ is of full rank, the components of $z^* = zC^\top$ are uncorrelated (and independent in ICA) and the model can be rewritten as $x = z^*{A^*}^\top$. Thus, the order, signs and scales of the source components cannot be determined. This means that, for any given unmixing matrix $W$, also $W^* = CW$ with $C \in \mathcal{C}$ is a solution.

As the scales of the latent components are not identifiable, one may simply assume that $\mathsf{COV}(z) = I_p$. Let then $\Sigma = \mathsf{COV}(x) = AA^\top$ denote the covariance matrix of $x$, and further let $\Sigma^{-1/2}$ be the symmetric matrix satisfying $\Sigma^{-1/2}\Sigma^{-1/2} = \Sigma^{-1}$. Then, for the standardized random variable $x_{st} = (x - \mu)\Sigma^{-1/2}$, we have that $z = x_{st}U^\top$ for some orthogonal $U$ (Miettinen, Taskinen, Nordhausen, and Oja 2015b, Theorem 1). Thus, the search for the unmixing matrix $W$ can be separated into finding the whitening (standardization) matrix $\Sigma^{-1/2}$ and the rotation matrix $U$. The unmixing matrix is then given by $W = U\Sigma^{-1/2}$.

In this paper, we describe the R package **JADE** (Nordhausen, Cardoso, Miettinen, Oja, Ollila, and Taskinen 2017) which offers several BSS methods covering all three major BSS models. In all of these methods, the whitening step is performed using the regular covariance matrix whereas the rotation matrix $U$ is found via joint diagonalization. The concepts of simultaneous and approximate joint diagonalization are recalled in Section 2, and several ICA, SOS and NSS methods based on diagonalization are described in Sections 3, 4 and 5, respectively. As performance indices are widely used to compare different BSS algorithms, we define some popular indices in Section 6. We also introduce the R package **BSSasymp** (Miettinen, Nordhausen, Oja, and Taskinen 2017) which includes functions for computing the asymptotic covariance matrices and their data-based estimates for most of the BSS estimators in the package **JADE**. Section 7 describes the R packages **JADE** and **BSSasymp**, and in Section 8 we illustrate the use of these packages via simulated and real data examples.

# 2. Simultaneous and approximate joint diagonalization

## 2.1. Simultaneous diagonalization of two symmetric matrices

Let $S_1$ and $S_2$ be two symmetric $p \times p$ matrices. If $S_1$ is positive definite, then there is a nonsingular $p \times p$ matrix $W$ and a diagonal $p \times p$ matrix $D$ such that

$$WS_1W^\top = I_p \quad \text{and} \quad WS_2W^\top = D.$$

If the diagonal values of $D$ are distinct, the matrix $W$ is unique up to a permutation and sign changes of the rows. Notice that the requirement that either $S_1$ or $S_2$ is positive definite is not necessary; there are more general results on simultaneous diagonalization of two symmetric matrices, see for example Golub and Van Loan (2002). However, for our purposes the assumption of positive definiteness is not too strong.

The simultaneous diagonalizer can be solved as follows. First solve the eigenvalue/eigenvector problem

$$S_1V^\top = V^\top\Lambda_1,$$

and define the inverse of the square root of $S_1$ as

$$S_1^{-1/2} = V^\top\Lambda_1^{-1/2}V.$$

Next solve the eigenvalue/eigenvector problem

$$(S_1^{-1/2}S_2(S_1^{-1/2})^\top)U^\top = U^\top\Lambda_2.$$

The simultaneous diagonalizer is then $W = US_1^{-1/2}$ and $D = \Lambda_2$.

## 2.2. Approximate joint diagonalization

Exact diagonalization of a set of symmetric $p \times p$ matrices $S_1, \ldots, S_K$, $K > 2$ is only possible if all matrices commute. As shown later in Sections 3, 4 and 5, in BSS this is, however, not the case for finite data and we need to perform approximate joint diagonalization, that is, we try to make $WS_KW^\top$ as diagonal as possible. In practice, we have to choose a measure of diagonality $M$, a function that maps a set of $p \times p$ matrices to $[0, \infty)$, and seek $W$ that minimizes

$$\sum_{k=1}^{K} M(WS_kW^\top).$$

Usually the measure of diagonality is chosen to be

$$M(V) = \|\text{off}(V)\|^2 = \sum_{i \neq j}(V)_{ij}^2,$$

where $\text{off}(V)$ has the same off-diagonal elements as $V$, and the diagonal elements are zero. In common principal component analysis for positive definite matrices, Flury (1984) used the measure

$$M(V) = \log\det(\text{diag}(V)) - \log\det(V),$$

where $\text{diag}(V) = V - \text{off}(V)$.

Obviously the sum of squares criterion is minimized by the trivial solution $W = 0$. The most popular method to avoid this solution is to diagonalize one of the matrices, then transform the rest of the $K-1$ matrices, and approximately diagonalize them requiring the diagonalizer to be orthogonal. To be more specific, suppose that $S_1$ is a positive definite $p \times p$ matrix. Then find $S_1^{-1/2}$ and denote $S_k^* = S_1^{-1/2} S_k (S_1^{-1/2})^\top$, $k = 2, \ldots, K$. Notice that in classical BSS methods, matrix $S_1$ is usually the covariance matrix, and the transformation is called whitening. Now if we measure the diagonality using the sum of squares of the off-diagonal elements, the approximate joint diagonalization problem is equivalent to finding an orthogonal $p \times p$ matrix $U$ that minimizes

$$\sum_{k=2}^{K} \|\mathrm{off}(U S_k^* U^\top)\|^2 = \sum_{k=2}^{K} \sum_{i \neq j} (U S_k^* U^\top)_{ij}^2.$$

Since the sum of squares remains the same when multiplied by an orthogonal matrix, we may equivalently maximize the sum of squares of the diagonal elements

$$\sum_{k=2}^{K} \|\mathrm{diag}(U S_k^* U^\top)\|^2 = \sum_{k=2}^{K} \sum_{i=1}^{p} (U S_k^* U^\top)_{ii}^2. \tag{3}$$

Several algorithms for orthogonal approximate joint diagonalization have been suggested, and in the following we describe two algorithms which are given in the R package **JADE**. For examples of nonorthogonal approaches, see R package **jointDiag** (Gouy-Pailler 2009) and references therein as well as Yeredor (2002).

The `rjd` algorithm uses Given's (or Jacobi) rotations to transform the set of matrices to a more diagonal form two rows and two columns at a time (Clarkson 1988). The Givens rotation matrix is given by

$$G(i,j,\theta) = \begin{pmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & \cos(\theta) & \cdots & -\sin(\theta) & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & \sin(\theta) & \cdots & \cos(\theta) & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{pmatrix}$$

In the `rjd` algorithm the initial value for the orthogonal matrix $U$ is $I_p$. First, the value of $\theta$ is computed using the elements $(S_k^*)_{11}$, $(S_k^*)_{12}$ and $(S_k^*)_{22}$, $k = 2, \ldots, K$, and matrices $U, S_2^*, \ldots, S_K^*$ are then updated by

$$U \leftarrow U G(1,2,\theta) \quad \text{and} \quad S_k^* \leftarrow G(1,2,\theta) S_k^* G(1,2,\theta), \quad k = 2, \ldots, K.$$

Similarly all pairs $i < j$ are gone through. When $\theta = 0$, the Givens rotation matrix is the identity and no more rotation is done. Hence, the convergence has been reached when $\theta$ is small for all pairs $i < j$. Based on vast simulation studies it seems that the solution of the `rjd` algorithm always maximizes the diagonality criterion (3).

In the deflation based joint diagonalization (`djd`) algorithm the rows of the joint diagonalizer are found one by one (Nordhausen, Gutch, Oja, and Theis 2012). Following the notations

above, assume that $S_2^*, \ldots, S_K^*$, $K \geq 2$, are the symmetric $p \times p$ matrices to be jointly diagonalized by an orthogonal matrix, and write the criterion (3) as

$$\sum_{k=2}^{K} \|\mathrm{diag}(US_k^*U^\top)\|^2 = \sum_{j=1}^{p}\sum_{k=2}^{K}(u_j S_k^* u_j^\top)^2, \tag{4}$$

where $u_j$ is the $j$th row of $U$. The sum (4) can then be approximately maximized by solving successively for each $j = 1, \ldots, p-1$, $u_j$ that maximizes

$$\sum_{k=2}^{K}(u_j S_k^* u_j^\top)^2 \tag{5}$$

under the constraint $u_r u_j^\top = \delta_{rj}$, $r = 1, \ldots, j-1$. Recall that $\delta_{rj} = 1$ if $r = j$ and zero otherwise.

The `djd` algorithm in the R package **JADE** is based on gradients, and to avoid stopping at local maxima, the initial value for each row is chosen from a set of random vectors so that criterion (5) is maximized in that set. The `djd` function also has an option to choose the initial values to be the eigenvectors of the first matrix $S_2^*$ which makes the function faster, but does not guarantee that the global maximum is reached. Recall that even if the algorithm finds the global maximum in every step, the solution only approximately maximizes the criterion (4).

In the `djd` function also criteria of the form

$$\sum_{k=2}^{K} |u_j S_k^* u_j^\top|^r, \ r > 0,$$

can be used instead of (5), and if all matrices are positive definite, also

$$\sum_{k=2}^{K} \log(u_j S_k^* u_j^\top).$$

The joint diagonalization plays an important role is BSS. In the next sections, we recall the three major BSS models, and corresponding separation methods which are based on the joint diagonalization. All these methods are included in the R package **JADE**.

## 3. Independent component analysis

The independent component model assumes that the source vector $z$ in model (1) has mutually independent components. Based on this assumption, the mixing matrix $A$ in (1) is not well-defined, therefore some extra assumptions are usually made. Common assumptions on the source variable $z$ in the IC model are:

**(IC1)** the source components are mutually independent,

**(IC2)** $\mathsf{E}(z) = 0$ and $\mathsf{E}(z^\top z) = I_p$,

**(IC3)** at most one of the components is Gaussian, and

**(IC4)** each source component is independent and identically distributed.

Assumption (IC2) fixes the variances of the components, and thus the scales of the rows of $A$. Assumption (IC3) is needed as, for multiple normal components, independence and uncorrelatedness are equivalent. Thus, any orthogonal transformation of normal components preserves the independence.

Classical ICA methods are often based on maximizing the non-Gaussianity of the components. This approach is motivated by the central limit theorem which, roughly speaking, says that the sum of random variables is more Gaussian than the summands. Several different methods to perform ICA are proposed in the literature. For general overviews, see for example Hyvärinen, Karhunen, and Oja (2001); Comon and Jutten (2010); Oja and Nordhausen (2012); Yu, Hu, and Xu (2014).

In the following, we review two classical ICA methods, FOBI and JADE, which utilize joint diagonalization when estimating the unmixing matrix. As the FOBI method is a special case of ICA methods based on two scatter matrices with so-called independence property (Oja, Sirkiä, and Eriksson 2006), we will first recall some related definitions.

### 3.1. Scatter matrix and independence property

Let $F_x$ denote the cumulative distribution function of a $p$-variate random vector $x$. A matrix valued functional $S(F_x)$ is called a scatter matrix if it is positive definite, symmetric and affine equivariant in the sense that $S(F_{Ax+b}) = AS(F_x)A^\top$ for all $x$, full rank $p \times p$ matrices $A$ and all $p$-variate vectors $b$.

Oja *et al.* (2006) noticed that the simultaneous diagonalization of any two scatter matrices with the independence property yields the ICA solution. The issue was further studied in Nordhausen, Oja, and Ollila (2008a). A scatter matrix $S(F_x)$ with the independence property is defined to be a diagonal matrix for all $x$ with independent components. An example of a scatter matrix with the independence property is the covariance matrix, but when it comes to most scatter matrices, they do not possess the independence property (for more details, see Nordhausen and Tyler 2015). However, it was noticed in Oja *et al.* (2006) that if the components of $x$ are independent and symmetric, then $S(F_x)$ is diagonal for any scatter matrix. Thus a symmetrized version of a scatter matrix $S_{\text{sym}}(F_x) = S(F_{x_1 - x_2})$, where $x_1$ and $x_2$ are independent copies of $x$, always has the independence property, and can be used to solve the ICA problem.

The affine equivariance of the matrices, which are used in the simultaneous diagonalization and approximate joint diagonalization methods, implies the affine equivariance of the unmixing matrix estimator. More precisely, if the unmixing matrices $W$ and $W^*$ correspond to $x$ and $x^* = xB^\top$, respectively, then $xW^\top = x^*{W^*}^\top$ (up to sign changes of the components) for all $p \times p$ full rank matrices $B$. This is a desirable property of an unmixing matrix estimator as it means that the separation result does not depend on the mixing procedure. It is easy to see that the affine equivariance also holds even if $S_2, \ldots, S_K$, $K \geq 2$, are only orthogonal equivariant.

### 3.2. FOBI

One of the first ICA methods, FOBI (fourth order blind identification) introduced by Cardoso (1989), uses simultaneous diagonalization of the covariance matrix and the matrix based on

the fourth moments,

$$S_1(F_x) = \mathsf{COV}(x) \quad \text{and} \quad S_2(F_x) = \frac{1}{p+2}\mathsf{E}[\|S_1^{-1/2}(x - \mathsf{E}(x))\|^2(x - \mathsf{E}(x))^\top(x - \mathsf{E}(x))],$$

respectively. Notice that both $S_1$ and $S_2$ are scatter matrices with the independence property. The unmixing matrix is the simultaneous diagonalizer $W$ satisfying

$$WS_1(F_x)W^\top = I_p \quad \text{and} \quad WS_2(F_x)W^\top = D,$$

where the diagonal elements of $D$ are the eigenvalues of $S_2(F_z)$ given by $\mathsf{E}[z_i^4] + p - 1$, $i = 1, \ldots, p$. Thus, for a unique solution, FOBI requires that the independent components have different kurtosis values. The statistical properties of FOBI are studied in Ilmonen, Nevalainen, and Oja (2010a) and Miettinen *et al.* (2015b).

### 3.3. JADE

The JADE (joint approximate diagonalization of eigenmatrices) algorithm (Cardoso and Souloumiac 1993) can be seen as a generalization of FOBI since both of them utilize fourth moments. For a recent comparison of these two methods, see Miettinen *et al.* (2015b). Contrary to FOBI, the kurtosis values do not have to be distinct in JADE. The improvement is gained by increasing the number of matrices to be diagonalized as follows. Define, for any $p \times p$ matrix $M$, the fourth order cumulant matrix as

$$C(M) = \mathsf{E}[(x_{st}Mx_{st}^\top)x_{st}^\top x_{st}] - M - M^\top - \mathrm{tr}(M)I_p,$$

where $x_{st}$ is a standardized variable. Notice that $C(I_p)$ is the matrix based on the fourth moments used in FOBI. Write then $E^{ij} = e_i^\top e_j,\ i, j = 1, \ldots, p$, where $e_i$ is a $p$-vector with the $i$th element one and others zero. In JADE (after the whitening) the matrices $C(E^{ij}),\ i, j = 1, \ldots, p$ are approximately jointly diagonalized by an orthogonal matrix. The rotation matrix $U$ thus maximizes the approximate joint diagonalization criterion

$$\sum_{i=1}^{p}\sum_{j=1}^{p} \|\mathrm{diag}(UC(E^{ij})U^\top)\|^2.$$

JADE is affine equivariant even though the matrices $C(E^{ij}),\ i, j = 1, \ldots, p$, are not orthogonal equivariant. If the eighth moments of the independent components are finite, then the vectorized JADE unmixing matrix estimate has a limiting multivariate normal distribution. For the asymptotic covariance matrix and a detailed discussion about JADE, see Miettinen *et al.* (2015b).

The JADE estimate jointly diagonalizes $p^2$ matrices. Hence its computational load grows quickly with the number of components. Miettinen, Nordhausen, Oja, and Taskinen (2013) suggested a quite similar, but faster method, called $k$-JADE which is computationally much simpler. The $k$-JADE method whitens the data using FOBI and then jointly diagonalizes

$$\{C(E^{ij}) :\ i, j = 1, \ldots, p,\ \text{with } |i - j| < k\}.$$

The value $k \leq p$ can be chosen by the user and corresponds basically to the guess of the largest multiplicity of identical kurtosis values of the sources. If $k$ is larger or equal to the largest multiplicity, then $k$-JADE and JADE seem to be asymptotically equivalent.

## 4. Second order source separation

In second order source separation (SOS) model, the source vectors compose a $p$-variate time series $z = (z_t)_{t=0,\pm1,\pm2,\ldots}$ that satisfies:

**(SOS1)** $\mathsf{E}(z_t) = 0$ and $\mathsf{E}(z_t^\top z_t) = I_p$, and

**(SOS2)** $\mathsf{E}(z_t^\top z_{t+\tau}) = D_\tau$ is diagonal for all $\tau = 1, 2, \ldots$

Above assumptions imply that the components of $z$ are weakly stationary and uncorrelated time series. In the following we will shortly describe two classical (SOS) methods, which yield affine equivariant unmixing matrix estimates.

The AMUSE (algorithm for multiple unknown signals extraction; Tong, Soon, Huang, and Liu 1990) uses the method of simultaneous diagonalization of two matrices. In AMUSE, the matrices to be diagonalized are the covariance matrix and the autocovariance matrix with chosen lag $\tau$, that is,

$$S_0(F_x) = \mathsf{COV}(x) \quad \text{and} \quad S_\tau(F_x) = \mathsf{E}[(x_t - \mathsf{E}(x_t))^\top (x_{t+\tau} - \mathsf{E}(x_t))].$$

The unmixing matrix $W_\tau$ then satisfies

$$W_\tau S_0(F_x) W_\tau^\top = I_p \quad \text{and} \quad W_\tau S_\tau(F_x) W_\tau^\top = D_\tau.$$

The requirement for distinct eigenvalues implies that the autocorrelations with the chosen lag need to be unequal for the source components. Notice that, as the population quantity $S_\tau(F_x)$ is symmetric, the estimate $\hat{W}_\tau$ is obtained by diagonalizing the sample covariance matrix and the symmetrized autocovariance matrix with lag $\tau$. The sample autocovariance matrix with lag $\tau$ is given by

$$\hat{S}_\tau(X) = \frac{1}{n-\tau} \sum_{t=1}^{n-\tau} (X_t - \bar{X}_t)^\top (X_{t+\tau} - \bar{X}_t),$$

and the symmetrized autocovariance matrix with lag $\tau$,

$$\hat{S}_\tau^S(X) = \frac{1}{2}(\hat{S}_\tau(X) + \hat{S}_\tau(X)^\top),$$

respectively.

It has been shown that the choice of the lag has a great impact on the performance of the AMUSE estimate (Miettinen, Nordhausen, Oja, and Taskinen 2012). However, without any preliminary knowledge of the uncorrelated components it is difficult to choose the best lag for the problem at hand. Cichocki and Amari (2002) simply recommend to start with $\tau = 1$, and check the diagonal elements of the estimate $\hat{D}$. If there are two almost equal values, another value for $\tau$ should be chosen.

Belouchrani, Abed-Meraim, Cardoso, and Moulines (1997) provide a natural approximate joint diagonalization method for the SOS model. In SOBI (second order blind identification) the data is whitened using the covariance matrix $S_0(F_x) = \mathsf{COV}(x)$. The $K$ matrices for rotation are then autocovariance matrices with distinct lags $\tau_1, \ldots, \tau_K$, that is, $S_{\tau_1}(F_x), \ldots, S_{\tau_K}(F_x)$. The use of different joint diagonalization methods yields estimates

with different properties. For details about the deflation-based algorithm (`djd`) in SOBI see Miettinen, Nordhausen, Oja, and Taskinen (2014), and for details about SOBI using the `rjd` algorithm see Miettinen, Illner, Nordhausen, Oja, Taskinen, and Theis (2016). General agreement seems to be that in most cases, the use of `rjd` in SOBI is preferable.

The problem of choosing the set of lags $\tau_1, \ldots, \tau_K$ for SOBI is not as important as the choice of lag $\tau$ for AMUSE. Among the signal processing community, $K = 12$ and $\tau_k = k$ for $k = 1, \ldots, K$, are conventional choices. Miettinen *et al.* (2014) argue that, when the deflation-based joint diagonalization is applied, one should rather take too many matrices than too few. The same suggestion applies to SOBI using the `rjd`. If the time series are linear processes, the asymptotic results in Miettinen *et al.* (2016) provide tools to choose the set of lags, see also Example 2 in Section 8.

## 5. Nonstationary source separation

The SOS model assumptions are sometimes considered to be too strong. The NSS model is a more general framework for cases where the data are ordered observations. In addition to the basic BSS model (1) assumptions, the following assumptions on the source components are made:

**(NSS1)** $\mathsf{E}(z_t) = 0$ for all $t$,

**(NSS2)** $\mathsf{E}(z_t^\top z_t)$ is positive definite and diagonal for all $t$,

**(NSS3)** $\mathsf{E}(z_t^\top z_{t+\tau})$ is diagonal for all $t$ and $\tau$.

Hence the source components are uncorrelated and they have a constant mean. However, the variances are allowed to change over time. Notice that this definition differs from the block-nonstationary model, where the time series can be divided into intervals so that the SOS model holds for each interval.

NSS-SD, NSS-JD and NSS-TD-JD are algorithms that take into account the nonstationarity of the variances. For the description of the algorithms define

$$S_{T,\tau}(F_x) = \frac{1}{|T| - \tau} \sum_{t \in T} \mathsf{E}[(x_t - \mathsf{E}(x_t))^\top (x_{t+\tau} - \mathsf{E}(x_t))],$$

where $T$ is a finite time interval and $\tau \in \{0, 1, \ldots\}$.

The NSS-SD unmixing matrix simultaneously diagonalizes $S_{T_1,0}(F_x)$ and $S_{T_2,0}(F_x)$, where $T_1,\ T_2 \subset [1, n]$ are separate time intervals. $T_1$ and $T_2$ should be chosen so that $S_{T_1,0}(F_x)$ and $S_{T_2,0}(F_x)$ are as different as possible.

The corresponding approximate joint diagonalization method is called NSS-JD. The data is whitened using the covariance matrix $S_{[1,n],0}(F_x)$ computed from all the observations. After whitening, the $K$ covariance matrices $S_{T_1,0}(F_x), \ldots, S_{T_K,0}(F_x)$ related to time intervals $T_1, \ldots, T_K$ are diagonalized with an orthogonal matrix.

Both NSS-SD and NSS-JD algorithms ignore the possible time dependence. Assume that the full time series can be divided into $K$ time intervals $T_1, \ldots, T_K$ so that, in each interval, the SOS model holds approximately. Then the autocovariance matrices within the intervals make sense, and the NSS-TD-JD algorithm is applicable. Again, the covariance matrix $S_{[1,n],0}(F_x)$

whitens the data. Now the matrices to be jointly diagonalized are $S_{T_i,\tau_j}(F_x)$, $i = 1, \ldots, K$, $j = 1, \ldots, L$. When selecting the intervals one should take into account the lengths of the intervals so that the random effect is not too large when the covariances and the autocovariances are computed. A basic rule among the signal processing community is to have 12 intervals if the data is large enough, or $K < 12$ intervals such that each interval contains at least 100 observations. Notice that NSS-SD and NSS-JD (as well as AMUSE and SOBI) are special cases of NSS-TD-JD. Naturally, NSS-SD, NSS-JD and NSS-TD-JD are all affine equivariant.

For further details on NSS methods see for example Choi and Cichocki (2000b,a); Choi, Cichocki, and Belouchrani (2001); Nordhausen (2014). Notice that asymptotic results are not yet available for any of these NSS methods.

## 6. BSS performance criteria

The performance of different BSS methods using real data is often difficult to evaluate since the true unmixing matrix is unknown. In simulations studies, however, the situation is different, and in the literature many performance indices have been suggested to measure the performance of different methods. For a recent overview see Nordhausen, Ollila, and Oja (2011), for example.

The package **JADE** contains several performance indices but in the following we will only introduce two of them. Both performance indices are functions of the so-called gain matrix, $\hat{G}$, which is a product of the unmixing matrix estimate $\hat{W}$ and the true mixing matrix, that is, $\hat{G} = \hat{W}A$. Since the order, the signs and the scales of the source components cannot be estimated, the gain matrix of an optimal estimate does not have to be identity, but equivalent to the identity in the sense that $\hat{G} = C$ for some $C \in \mathcal{C}$, where $\mathcal{C}$ is given in (2).

The Amari error (Amari, Cichocki, and Yang 1996) is defined as

$$AE(\hat{G}) = \frac{1}{2p(p-1)} \left( \sum_{i=1}^{p} \left( \sum_{j=1}^{p} \frac{|\hat{g}_{ij}|}{\max_h |\hat{g}_{ih}|} - 1 \right) + \sum_{j=1}^{p} \left( \sum_{i=1}^{p} \frac{|\hat{g}_{ij}|}{\max_h |\hat{g}_{hj}|} - 1 \right) \right),$$

where $\hat{g}_{ij}$ denotes the $ij$th element of $\hat{G}$. The range of the Amari error values is $[0, 1]$, and a small value corresponds to a good separation performance. The Amari error is not scale invariant. Therefore, when different algorithms are compared, the unmixing matrices should be scaled in such a way that the corresponding rows of different matrices are of equal length.

The minimum distance index (Ilmonen, Nordhausen, Oja, and Ollila 2010b) is defined as

$$MD(\hat{G}) = \frac{1}{\sqrt{p-1}} \inf_{C \in \mathcal{C}} \|C\hat{G} - I_p\|,$$

where $\|\cdot\|$ is the matrix (Frobenius) norm and $\mathcal{C}$ is defined in (2). Also the MD index is scaled to have a maximum value 1, and $MD(\hat{G}) = 0$ if and only if $\hat{G} \in \mathcal{C}$. The MD index is affine invariant. The statistical properties of the index are thoroughly studied in Ilmonen *et al.* (2010b) and Ilmonen, Nordhausen, Oja, and Ollila (2012).

A feature that makes the minimum distance index especially attractive in simulation studies is that its value can be related to the asymptotic covariance matrix of an estimator $\hat{W}$. If $\hat{W} \to A^{-1}$ and $\sqrt{n}\,\mathrm{vec}(\hat{W}A - I_p) \to N_{p^2}(0, \Sigma)$, which is for example the case for FOBI, JADE,

AMUSE and SOBI, then the limiting distribution of $n(p-1)MD(\hat{G})^2$ has as expected value

$$\text{tr}\left((I_{p^2} - D_{p,p})\Sigma(I_{p^2} - D_{p,p})\right), \tag{6}$$

where $D_{p,p} = \sum_{i=1}^{p} e_i^\top e_i \otimes e_i^\top e_i$, with $\otimes$ denoting the Kronecker product and $e_i$ a $p$-vector with $i$th element one and others zero.

Notice that $\text{tr}\left((I_{p^2} - D_{p,p})\Sigma(I_{p^2} - D_{p,p})\right)$ is the sum of the off-diagonal elements of $\Sigma$ and therefore a natural measure of the variation of the unmixing matrix estimate $\hat{W}$. We will make use of this relationship later in one of our examples.

## 7. Functionality of the packages

The package **JADE** is freely available from the Comprehensive R Archive Network (CRAN) at http://CRAN.R-project.org/package=JADE and comes under the GNU General Public License (GPL) 2.0 or higher.

The main functions of the package implement the blind source separation methods described in the previous sections. The function names are self explanatory being `FOBI`, `JADE` and `k_JADE` for ICA, `AMUSE` and `SOBI` for SOS and `NSS.SD`, `NSS.JD` and `NSS.TD.JD` for NSS. All functions usually take as an input either a numerical matrix `X` or as alternative a multivariate time series object of class '`ts`'. The functions have method appropriate arguments like for example which lags to choose for `AMUSE` and `SOBI`.

All functions return an object of the S3 class '`bss`' which contains at least an object `W`, which is the estimated unmixing matrix, and `S` containing the estimated (and centered) sources. Depending on the chosen function also other information is stored. The methods available for the class are

- `print`: which prints basically all information except the sources `S`.
- `coef`: which extracts the unmixing matrix `W`.
- `plot`: which produces a scatter plot matrix of `S` using `pairs` for ICA methods and a multivariate time series plot using the `plot` method for '`ts`' objects for other BSS methods.

To extract the sources `S` the helper function `bss.components` can be used.

The functions which use joint approximate diagonalization of several matrices provide the user an option to choose the method for joint diagonalization from the list below.

- `djd`: for deflation-based joint diagonalization.
- `rjd`: for joint diagonalization using Givens rotations.
- `frjd`: which is basically the same as `rjd`, but has less options and is much faster because it is implemented in C.

From our experience the function `frjd`, when appropriate, seems to obtain the best results.

In addition, the **JADE** package provides two other functions for joint diagonalization. The function `FG` is designed for diagonalization of real positive-definite matrices and `cjd` is the generalization of `rjd` to the case of complex matrices. For details about all functions for joint diagonalization see also their help pages. More functions for joint diagonalization are also available in the R package **jointDiag**.

To evaluate the performance of BSS methods using simulation studies, performance indices are needed. The package provides for this purpose the functions `amari_error`, `ComonGAP`, `MD` and `SIR`. Our personal favorite is the `MD` function which implements the minimum distance index described in Section 6. For further details on all the functions see their help pages and the references therein.

For ICA, many alternative methods are implemented in other R packages. Examples include **fastICA** (Marchini, Heaton, and Ripley 2013), **fICA** (Miettinen, Nordhausen, Oja, and Taskinen 2015a), **mlica2** (Teschendorff 2012), **PearsonICA** (Karvanen 2009) and **ProDenICA** (Hastie and Tibshirani 2010). None of these ICA methods uses joint diagonalization in estimation. Two slightly overlapping packages with **JADE** are **ICS** (Nordhausen, Oja, and Tyler 2008b) which provides a generalization of the FOBI method, and **ica** (Helwig 2015) which includes the JADE algorithm. In current practice JADE and fastICA (implemented for example in the packages **fastICA** and **fICA**) seem to be the most often used ICA methods. Other newer ICA methods, as for example ICA via product density estimation as provided in the package **ProDenICA**, are often computationally very intensive as the sample size is usually high in typical ICA applications. To the best of our knowledge there are currently no other R packages for SOS or NSS available.

Many methods included in the **JADE** package are also available in the MATLAB (The MathWorks Inc. 2014) toolbox **ICALAB** (Cichocki, Amari, Siwek, Tanaka, Phan, and others 2014) which accompanies the book of Cichocki and Amari (2002). A collection of links to JADE implementations for real and complex values in different languages like MATLAB, C and Python as well as some joint diagonalization functions for MATLAB are available on J.-F. Cardoso's homepage (`http://perso.telecom-paristech.fr/~cardoso/guidesepsou.html`).

The package **BSSasymp** is freely available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=BSSasymp` and comes under the GNU General Public License (GPL) 2.0 or higher.

There are two kinds of functions in the package. The first set of functions compute the asymptotic covariance matrices of the vectorized mixing and unmixing matrix estimates under different BSS models. The others estimate the covariance matrices based on a data matrix. The package **BSSasymp** includes functions for several estimators implemented in package **JADE**. These are FOBI and JADE in the IC model and AMUSE, deflation-based SOBI and regular SOBI in the SOS model. The asymptotic covariance matrices for FOBI and JADE estimates are computed using the results in Miettinen *et al.* (2015b). For the limiting distributions of AMUSE and SOBI estimates, see Miettinen *et al.* (2012) and Miettinen *et al.* (2016), respectively.

Functions `ASCOV_FOBI` and `ASCOV_JADE` compute the theoretical values for covariance matrices. The argument `sdf` is the vector of source density functions standardized to have mean zero and variance equal to one, `supp` is a two column matrix, whose rows give the lower and the upper limits used in numerical integration for each source component and `A` is the mixing matrix. The corresponding functions `ASCOV_SOBIdefl` and `ASCOV_SOBI` in the SOS

model take as input the matrix `psi`, which gives the MA coefficients of the source time series, the vector of integers `taus` for the lags, a matrix of fourth moments of the innovations `Beta` (default value is for Gaussian innovations) and the mixing matrix `A`.

Functions `ASCOV_FOBI_est`, `ASCOV_JADE_est`, `ASCOV_SOBIdefl_est` and `ASCOV_SOBI_est` can be used for approximating the covariance matrices of the estimates. They are based on asymptotic results, and therefore the sample size should not be very small. Argument `X` can be either the observed data or estimated source components. When argument `mixed` is set to `TRUE`, `X` is considered as observed data and the unmixing matrix is first estimated using the method of interest. The estimation of the covariance matrix of the SOBI estimate is also based on the assumption that the time series are stationary linear processes. If the time series are Gaussian, then the asymptotic variances and covariances depend only on the autocovariances of the source components. Argument `M` gives the number of autocovariances to be used in the approximation of the infinite sums of autocovariances. Thus, `M` should be the largest lag for which any of the source time series has non-zero autocovariance. In the non-Gaussian case the coefficients of the linear processes need to be computed. In functions `ASCOV_SOBIdefl_est` and `ASCOV_SOBI_est`, ARMA parameter estimation is used and arguments `arp` and `maq` fix the order of ARMA series, respectively. There are also faster functions `ASCOV_SOBIdefl_estN` and `ASCOV_SOBI_estN`, which assume that the time series are Gaussian and do not estimate the MA coefficients. The argument `taus` is to define the lags of the SOBI estimate.

All functions for the theoretical asymptotic covariance matrices return lists with five components. `A` and `W` are the mixing and unmixing matrices and `COV_A` and `COV_W` are the corresponding asymptotic covariance matrices. In simulations studies where the MD index is used as performance criterion, the sum of the variance of the off-diagonal values is of interest (recall Section 6 for details). This sum is returned as object `EMD` in the list.

The functions for the estimated asymptotic covariance matrices return similar lists as their theoretic counterparts excluding the component `EMD`.

## 8. Examples

In this section we provide four examples to demonstrate how to use the main functions in the packages **JADE** and **BSSasymp**. In Section 8.1 we show how different BSS methods can be compared using an artificial data set. Section 8.2 demonstrates how the package **BSSasymp** can help in selecting the best method for the source separation. In Section 8.3 we show how a typical simulation study for the comparison of BSS estimators can be performed using the packages **JADE** and **BSSasymp**, and finally, in Section 8.4 a real data example is considered. In these examples the dimension of the data is relatively small, but for example in Joyce, Gorodnitsky, and Kutas (2004) SOBI has been successfully applied to analyze EEG data where the electrical activity of the brain is measured by 128 sensors on the scalp. As mentioned earlier, computation of the JADE estimate for such high-dimensional data is demanding because of the large number of matrices and the use of $k$-JADE is recommended then.

In the examples we use the option `options(digits = 4)` in R 3.3.2 together with the packages **JADE** 2.0-0, **BSSasymp** 1.2-0 and **tuneR** 1.3.1 (Ligges, Krey, Mersmann, and Schnackenberg 2016) for the output. Random seeds (when applicable) are provided for reproducibility of examples.

### 8.1. Example 1

A classical example of the application of BSS is the so-called cocktail party problem. To separate the voices of $p$ speakers, we need $p$ microphones in different parts of the room. The microphones record the mixtures of all $p$ speakers and the goal is then to recover the individual speeches from the mixtures. To illustrate the problem the **JADE** package contains in its subfolder `datafiles` three audio files which are often used in BSS[1]. For demonstration purpose we mix the audio files and try to recover the original sounds again. The cocktail party problem data can be created using the packages

```
R> library("JADE")
R> library("tuneR")
```

and loading the audio files as follows

```
R> S1 <- readWave(system.file("datafiles/source5.wav", package = "JADE"))
R> S2 <- readWave(system.file("datafiles/source7.wav", package = "JADE"))
R> S3 <- readWave(system.file("datafiles/source9.wav", package = "JADE"))
```

We attach a noise component in the data, scale the components to have unit variances, and then mix the sources with a mixing matrix. The components of a mixing matrix were generated from a standard normal distribution.

```
R> set.seed(321)
R> NOISE <- noise("white", duration = 50000)
R> S <- cbind(S1@left, S2@left, S3@left, NOISE@left)
R> S <- scale(S, center = FALSE, scale = apply(S, 2, sd))
R> St <- ts(S, start = 0, frequency = 8000)
R> p <- 4
R> A <- matrix(runif(p^2, 0, 1), p, p)
R> A

       [,1]     [,2]   [,3]   [,4]
[1,] 0.1989 0.066042 0.7960 0.4074
[2,] 0.3164 0.007432 0.4714 0.7280
[3,] 0.1746 0.294247 0.3068 0.1702
[4,] 0.7911 0.476462 0.1509 0.6219

R> X <- tcrossprod(St, A)
R> Xt <- as.ts(X)
```

Note that the mixed sound signal data `X` is for convenience also provided as the data set `CPPdata` in the **JADE** package.

Figures 1 and 2 show the original sound sources and mixed sources, respectively. These are obtained using the code

[1]The files are originally downloaded from http://research.ics.aalto.fi/ica/cocktail/cocktail_en.cgi and the authors are grateful to Docent Ella Bingham for giving the permission to use the audio files.

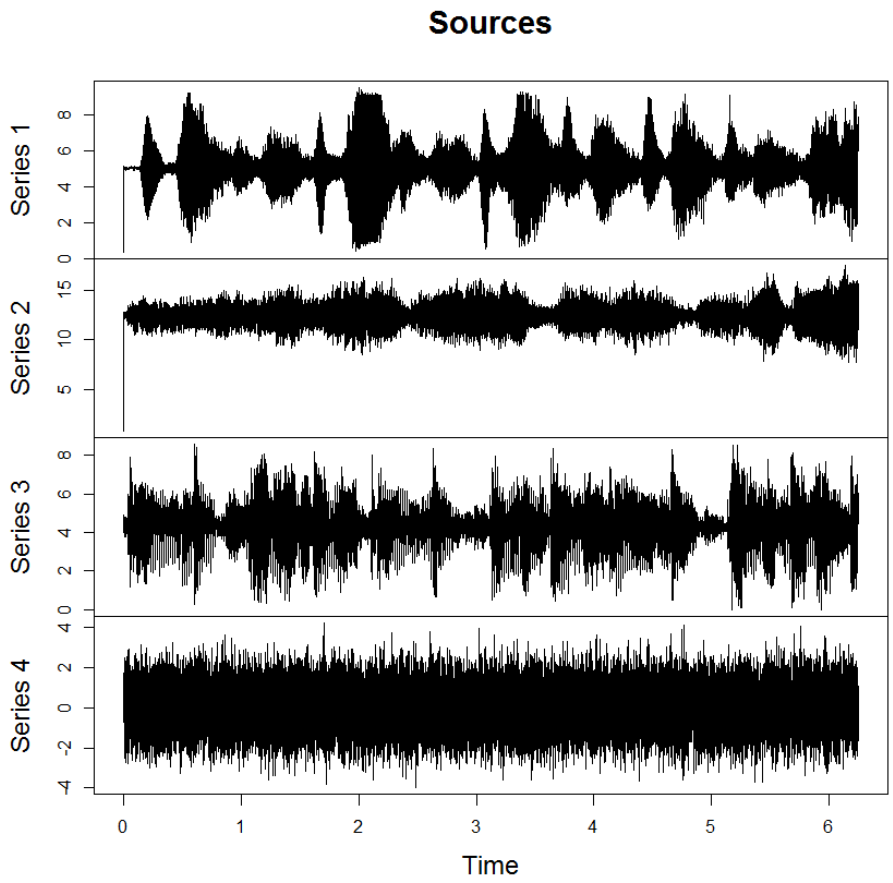


Figure 1: Original sound and noise signals.

```
R> plot(St, main = "Sources")
R> plot(Xt, main = "Mixtures")
```

The package **tuneR** can play wav files directly from R if a media player is initialized using the function `setWavPlayer`. Assuming that this has been done, the four mixtures can be played using the code

```
R> x1 <- normalize(Wave(left = X[, 1], samp.rate = 8000, bit = 8),
+    unit = "8")
R> x2 <- normalize(Wave(left = X[, 2], samp.rate = 8000, bit = 8),
+    unit = "8")
R> x3 <- normalize(Wave(left = X[, 3], samp.rate = 8000, bit = 8),
+    unit = "8")
R> x4 <- normalize(Wave(left = X[, 4], samp.rate = 8000, bit = 8),
+    unit = "8")
R> play(x1)
R> play(x2)
R> play(x3)
R> play(x4)
```

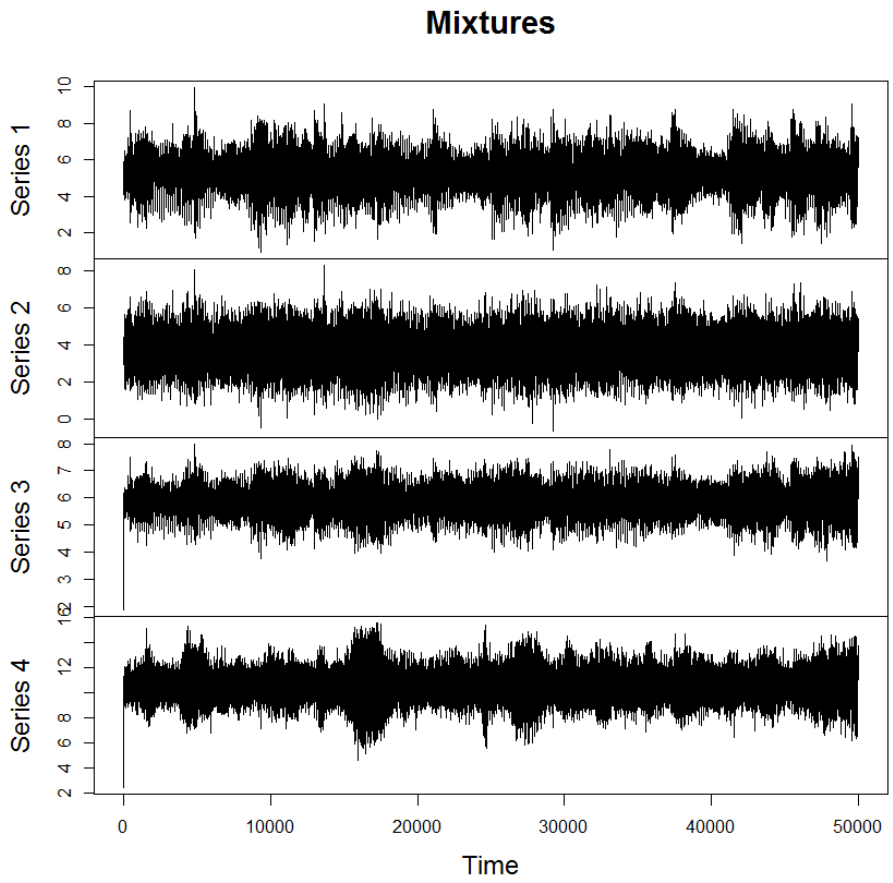


Figure 2: Mixed sound signals.

To demonstrate the use of BSS methods, assume now that we have observed the mixture of unknown source signals plotted in Figure 2. The aim is then to estimate the original sound signals based on this observed data. The question is then, which method to use. Based on Figure 2, the data are neither iid nor second order stationary. Nevertheless, we first apply JADE, SOBI and NSSTDJD with their default settings:

```
R> jade <- JADE(X)
R> sobi <- SOBI(Xt)
R> nsstdjd <- NSS.TD.JD(Xt)
```

All three objects are then of class '`bss`' and for demonstration purposes we look at the output of the call to `SOBI`.

```
R> sobi
```

```
W :
       [,1]    [,2]    [,3]     [,4]
[1,]  1.931 -0.9493 -0.2541 -0.08017
[2,] -2.717  1.1377  5.8263 -1.14549
```

```
[3,] -3.093  2.9244  4.7697 -2.70582
[4,] -2.709  3.3365  2.4661 -1.19771

k :
 [1]  1  2  3  4  5  6  7  8  9 10 11 12

method :
[1] "frjd"
```

The SOBI output tells us that the autocovariance matrices with the lags listed in k have been jointly diagonalized with the method "frjd" yielding the unmixing matrix estimate W. If however another set of lags would be preferred, this can be achieved as follows:

```
R> sobi2 <- SOBI(Xt, k = c(1, 2, 5, 10, 20))
```

In such an artificial framework, where the mixing matrix is available, one can compute the product $\hat{W}A$ in order to see if it is close to a matrix with only one non-zero element per row and column.

```
R> round(coef(sobi) %*% A, 4)

        [,1]    [,2]    [,3]    [,4]
[1,] -0.0241  0.0075  0.9995  0.0026
[2,] -0.0690  0.9976 -0.0115  0.0004
[3,] -0.9973 -0.0683 -0.0283 -0.0025
[4,]  0.0002  0.0009 -0.0074  1.0000
```

The matrix $\hat{W}A$ has exactly one large element on each row and column which expresses that the separation was successful. A more formal way to evaluate the performance is to use a performance index. We now compare all four methods using the minimum distance index.

```
R> c(jade = MD(coef(jade), A), sobi = MD(coef(sobi), A),
+    sobi2 = MD(coef(sobi2), A), nsstdjd = MD(coef(nsstdjd), A))

   jade    sobi   sobi2 nsstdjd
0.07505 0.06072 0.03372 0.01388
```

MD indices show that NSSTDJD performs best and that JADE is the worst method here. This result is in agreement with how well the data meets the assumptions of each method. The SOBI with the second set of lags is better than the default SOBI. In Section 8.2 we show how the package **BSSasymp** can be used to select a good set of lags.

To play the sounds recovered by NSSTDJD, one can use the function bss.components to extract the estimated sources and convert them back to audio.

```
R> Z.nsstdjd <- bss.components(nsstdjd)
R> NSSTDJDwave1 <- normalize(Wave(left = as.numeric(Z.nsstdjd[, 1]),
+    samp.rate = 8000, bit = 8), unit = "8")
```

```
R> NSSTDJDwave1 <- normalize(Wave(left = as.numeric(Z.nsstdjd[, 2]),
+    samp.rate = 8000, bit = 8), unit = "8")
R> NSSTDJDwave1 <- normalize(Wave(left = as.numeric(Z.nsstdjd[, 3]),
+    samp.rate = 8000, bit = 8), unit = "8")
R> NSSTDJDwave1 <- normalize(Wave(left = as.numeric(Z.nsstdjd[, 4]),
+    samp.rate = 8000, bit = 8), unit = "8")
R> play(NSSTDJDwave1)
R> play(NSSTDJDwave2)
R> play(NSSTDJDwave3)
R> play(NSSTDJDwave4)
```

### 8.2. Example 2

We continue with the cocktail party data of Example 1 and show how the package **BSSasymp** can be used to select the lags for the SOBI method. The asymptotic results of Miettinen *et al.* (2016) are utilized in order to estimate the asymptotic variances of the elements of the SOBI unmixing matrix estimate $\hat{W}$ with different sets of lags. Our choice for the objective function to be minimized, with respect to the set of lags, is the sum of the estimated variances (see also Section 6). The number of different sets of lags is practically infinite. In this example we consider the following seven sets:

(i) 1 (AMUSE),

(ii) 1–3,

(iii) 1–12,

(iv) 1, 2, 5, 10, 20,

(iv) 1–50,

(v) 1–20, 25, 30, . . . , 100,

(vi) 11–50.

For the estimation of the asymptotic variances, we assume that the time series are stationary linear processes. Since we are not interested in the exact values of the variances, but wish to rank different estimates based on their performance measured by the sum of the limiting variances, we select the function `ASCOV_SOBI_estN` which assumes Gaussianity of the time series. Notice also that the effect of the non-Gaussianity seems to be rather small, see Miettinen *et al.* (2012). Now the user only needs to choose the value of `M`, the number of autocovariances to be used in the estimation. The value of `M` should be such that all lags with non-zero autocovariances are included, and the estimation of such autocovariances is still reliable. We choose `M = 1000`.

```
R> library("BSSasymp")
R> ascov1 <- ASCOV_SOBI_estN(Xt, taus = 1, M = 1000)
R> ascov2 <- ASCOV_SOBI_estN(Xt, taus = 1:3, M = 1000)
R> ascov3 <- ASCOV_SOBI_estN(Xt, taus = 1:12, M = 1000)
```

```
R> ascov4 <- ASCOV_SOBI_estN(Xt, taus = c(1, 2, 5, 10, 20), M = 1000)
R> ascov5 <- ASCOV_SOBI_estN(Xt, taus = 1:50, M = 1000)
R> ascov6 <- ASCOV_SOBI_estN(Xt, taus = c(1:20, (5:20) * 5), M = 1000)
R> ascov7 <- ASCOV_SOBI_estN(Xt, taus = 11:50, M = 1000)
```

The estimated asymptotic variances of the first estimate are now the diagonal elements of `ascov1$COV_W`. Since the true mixing matrix $A$ is known, it is also possible to use the MD index to find out how well the estimates perform. We can thus check whether the minimization of the sum of the limiting variances really yields a good estimate.

```
R> SumVar <- cbind("(i)" = sum(diag(ascov1$COV_W)),
+    "(ii)" = sum(diag(ascov2$COV_W)), "(iii)" = sum(diag(ascov3$COV_W)),
+    "(iv)" = sum(diag(ascov4$COV_W)), "(v)" = sum(diag(ascov5$COV_W)),
+    "(vi)" = sum(diag(ascov6$COV_W)), "(vii)" = sum(diag(ascov7$COV_W)))
R> MDs <- cbind("(i)" = MD(ascov1$W, A), "(ii)" = MD(ascov2$W, A),
+    "(iii)" = MD(ascov3$W, A), "(iv)" = MD(ascov4$W, A),
+    "(v)" = MD(ascov5$W, A), "(vi)" = MD(ascov6$W,A ),
+    "(vii)" = MD(ascov7$W, A))
R> SumVar

     (i)   (ii)  (iii)    (iv)    (v)    (vi)  (vii)
[1,] 363 0.1282 0.1362 0.08217 0.0756 0.06798 0.1268

R> MDs

       (i)    (ii)   (iii)    (iv)     (v)    (vi)  (vii)
[1,] 0.433 0.03659 0.06072 0.03372 0.01242 0.01231 0.0121
```

The variance estimates indicate that the lag one alone is not sufficient. Sets (iv), (v) and (vi) give the smallest sums of the variances. The minimum distance index values show that (i) really is the worst set here and that set (vi), whose estimated sum of asymptotic variances was the smallest, is a good choice here, even though set (vii) has slightly smaller minimum distance index value. Hence in a realistic data only situation, where performance indices cannot be computed, the sum of the variances can provide a way to select a good set of lags for the SOBI method.

### 8.3. Example 3

In simulation studies usually several estimators are compared and it is of interest to study which of the estimators performs best under the given model and also how fast the estimators converge to their limiting distributions. In the following we will perform a simulation study similar to that of Miettinen *et al.* (2016) and compare the performances of FOBI, JADE and 1-JADE using the package **BSSasymp**.

Consider the ICA model where the three source component distributions are exponential, uniform and normal distributions, all of them centered and scaled to have unit variances. Due to the affine equivariance of the estimators, the choice of the mixing matrix does not affect the performances, and we can choose $A = I_3$ for simplicity.

We first create a function `ICAsim` which generates the data and then computes the MD indices using the unmixing matrices estimated with the three ICA methods. The arguments in `ICAsim` are a vector of different sample sizes (`ns`) and the number of repetitions (`repet`). The function then returns a data frame with the variables `N`, `fobi`, `jade` and `kjade`, which includes the used sample size and the obtained MD index value for each run and for the three different methods.

```
R> library("JADE")
R> library("BSSasymp")
R> ICAsim <- function(ns, repet) {
+    M <- length(ns) * repet
+    MD.fobi <- numeric(M)
+    MD.jade <- numeric(M)
+    MD.1jade <- numeric(M)
+    A <- diag(3)
+    row <- 0
+    for (j in ns) {
+      for (i in 1:repet) {
+        row <- row + 1
+        x1 <- rexp(j) - 1
+        x2 <- runif(j, -sqrt(3), sqrt(3))
+        x3 <- rnorm(j)
+        X <- cbind(x1, x2, x3)
+        MD.fobi[row] <- MD(coef(FOBI(X)), A)
+        MD.jade[row] <- MD(coef(JADE(X)), A)
+        MD.1jade[row] <- MD(coef(k_JADE(X, k = 1)), A)
+      }
+    }
+    RES <- data.frame(N = rep(ns, each = repet), fobi = MD.fobi,
+      jade = MD.jade, kjade = MD.1jade)
+    RES
+  }
```

For each of the sample sizes, 250, 500, 1000, 2000, 4000, 8000, 16000 and 32000, we then generate 2000 repetitions. Notice that this simulation will take a while.

```
R> set.seed(123)
R> N <- 2^((-2):5) * 1000
R> MDs <- ICAsim(ns = N, repet = 2000)
```

Besides the finite sample performances of different methods, we are interested in seeing how quickly the estimators converge to their limiting distributions. The relationship between the minimum distance index and the asymptotic covariance matrix of the unmixing matrix estimate was described in Section 6. To compute (6) we first compute the asymptotic covariance matrices of the unmixing matrix estimates $\hat{W}$. Since all three independent components in the model have finite eighth moments, all three estimates have a limiting multivariate normal distribution (Ilmonen *et al.* 2010a; Miettinen *et al.* 2015b). The functions `ASCOV_FOBI` and

`ASCOV_JADE` compute the asymptotic covariance matrices of the corresponding unmixing matrix estimates $\hat{W}$ and the mixing matrix estimates $\hat{W}^{-1}$. As arguments, one needs the source density functions standardized so that the expected value is zero and the variance equals to one, and the support of each density function. The default value for the mixing matrix is the identity matrix.

```
R> f1 <- function(x) { exp(-x - 1) }
R> f2 <- function(x) { rep(1 / (2 * sqrt(3)), length(x)) }
R> f3 <- function(x) { exp(-(x)^2 / 2) / sqrt(2 * pi) }
R> support <- matrix(c(-1, -sqrt(3), -Inf, Inf, sqrt(3), Inf), nrow = 3)
R> fobi <- ASCOV_FOBI(sdf = c(f1, f2, f3), supp = support)
R> jade <- ASCOV_JADE(sdf = c(f1, f2, f3), supp = support)
```

Let us next look at the simulation results concerning the FOBI method in more detail. First notice that the rows of the FOBI unmixing matrices are ordered according to the kurtosis values of resulting independent components. Since the source distributions `f1`, `f2` and `f3` are not ordered accordingly, the unmixing matrix `fobi$W` is different from the identity matrix.

```
R> fobi$W
```

```
     [,1] [,2] [,3]
[1,]    1    0    0
[2,]    0    0    1
[3,]    0    1    0
```

Object `fobi$COV_W` is the asymptotic covariance matrix of the vectorized unmixing matrix estimate $\mathrm{vec}(\hat{W})$.

```
R> fobi$COV_W
```

```
      [,1]   [,2]   [,3]   [,4]   [,5] [,6]   [,7] [,8]   [,9]
 [1,]    2  0.000  0.000  0.000  0.000  0.0  0.000  0.0  0.000
 [2,]    0  6.189  0.000  0.000  0.000  0.0 -4.689  0.0  0.000
 [3,]    0  0.000  4.217 -3.037  0.000  0.0  0.000  0.0  0.000
 [4,]    0  0.000 -3.037  3.550  0.000  0.0  0.000  0.0  0.000
 [5,]    0  0.000  0.000  0.000 11.151  0.0  0.000  0.0  2.349
 [6,]    0  0.000  0.000  0.000  0.000  0.2  0.000  0.0  0.000
 [7,]    0 -4.689  0.000  0.000  0.000  0.0  5.189  0.0  0.000
 [8,]    0  0.000  0.000  0.000  0.000  0.0  0.000  0.5  0.000
 [9,]    0  0.000  0.000  0.000  2.349  0.0  0.000  0.0 10.151
```

The diagonal elements of `fobi$COV_W` are the asymptotic variances of $(\hat{W})_{11}$,$(\hat{W})_{22}$,...,$(\hat{W})_{pp}$, respectively, and the value $-3.037$, for example, in `fobi$COV_W` is the asymptotic covariance of $(\hat{W})_{31}$ and $(\hat{W})_{12}$.

To make use of the relationship between the minimum distance index and the asymptotic covariance matrices, we need to extract the asymptotic variances of the off-diagonal elements of such $\hat{W}A$ that converges to $I_3$. In fact, these variances are the second, third, fourth, fifth, seventh and ninth diagonal element of `fobi$COV_W`, but there is also an object `fobi$EMD`, which directly gives the sum of the variances as given in (6).

```
R> fobi$EMD
```

```
[1] 40.45
```

The corresponding value for JADE can be obtained as follows.

```
R> jade$EMD
```

```
[1] 23.03
```

Based on these results we can conclude that for this ICA model, the theoretically best separation method is JADE. The value $n(p-1)MD(\hat{G})^2$ for JADE should converge to 23.03 and that for FOBI to 40.45. Since all three components have different kurtosis values, 1-JADE is expected to have the same limiting behavior as JADE.

To compare the theoretical values to their finite sample counterparts, we next compute the average values of $n(p-1)MD(\hat{G})^2$ for each sample size and each estimator, and plot them together with their limiting expected values in Figure 3.

```
R> meanMDs <- aggregate(MDs[, 2:4]^2, list(N = MDs$N), mean)
R> MmeansMDs <- 2 * meanMDs[, 1] * meanMDs[, 2:4]
R> ylabel <- expression(paste("n(p-1)ave", (hat(D)^2)))
R> par(mar = c(4, 5, 0, 0) + 0.1)
R> matplot(N, MmeansMDs, pch = c(15, 17, 16), ylim = c(0, 60),
+    ylab =  ylabel, log = "x", xlab = "n", cex = c(1.5, 1.6, 1.2),
+    col = c(1, 2, 4), xaxt = "n")
R> axis(1, N)
R> abline(h = fobi$EMD, lty = 1, lwd = 2)
R> abline(h = jade$EMD, lty = 2, col = 4, lwd = 2)
R> legend("topright", c("FOBI", "JADE", "1-JADE"), lty = c(1, 2, 0),
+    pch = 15:17, col = c(1, 4, 2), bty = "n", pt.cex = c(1.5, 1.2, 1.6),
+    lwd = 2)
```

Figure 3 supports the fact that JADE and 1-JADE are asymptotically equivalent. For small sample sizes the finite sample performance of JADE is slightly better than that of 1-JADE. The average of squared minimum distance values of JADE seem to converge faster to its expected value than those of FOBI.

### 8.4. Example 4

So far we have considered examples where the true sources and the mixing matrix have been known. In our last example we use a real data set which includes electrocardiography (ECG) recordings of a pregnant woman. ECG measures the electrical potential, generated by the heart muscle, from the body surface. The electrical activity produced by the heart beats of a fetus can then be detected by measuring the potential on the mother's skin. As the measured signals are mixtures of the fetus's and the mother's heart beats, the goal is to use the BSS method to separate these two heart beats as well as some possible artifacts from each other. In this context it is useful to know that a fetus's heart is supposed to beat faster than that of

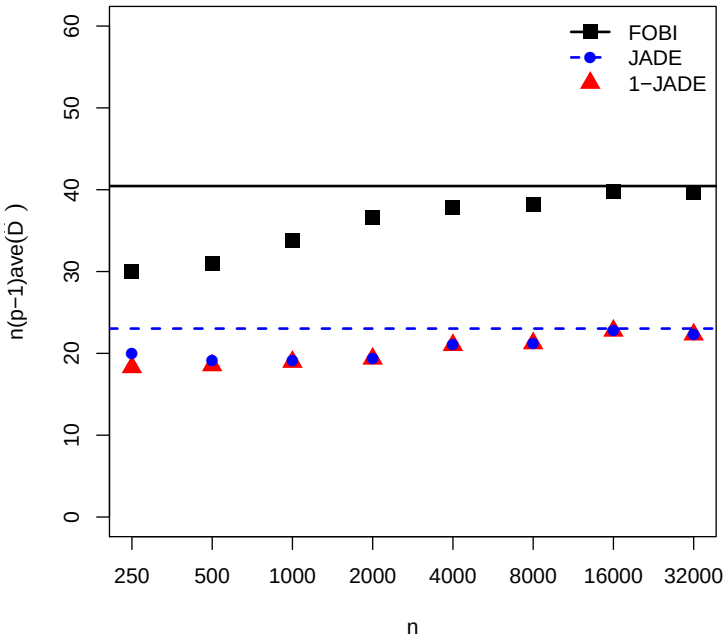


Figure 3: Simulation results based on 2000 repetitions. The dots give the average values of $n(p-1)MD(\hat{G})^2$ for each sample size, and the horizontal lines are the expected values of the limiting of the limiting distributions of $n(p-1)MD(\hat{G})^2$ for the FOBI method and the two JADE methods.

the mother. For a more detail discussion on the data and of the use of BSS in this context, see De Lathauwer, De Moor, and Vandewalle (1995).

In this ECG recording, eight sensors have been placed on the skin of the mother, the first five in the stomach area and the other three in the chest area. The data was obtained as `foetal_ecg.dat` from `http://homes.esat.kuleuven.be/~smc/daisy/daisydata.html` and is also provided in the supplementary files (with kind permission from Prof. Lieven De Lathauwer).

We first load the data assuming it is in the working directory and plot it in Figure 4.

```
R> library("JADE")
R> library("BSSasymp")
R> dataset <- matrix(scan("foetal_ecg.dat"), 2500, 9, byrow = TRUE)

Read 22500 items

R> X <- dataset[, 2:9]
R> plot.ts(X, nc = 1, main = "Data")
```

Figure 4 shows that the mother's heartbeat is clearly the main element in all of the signals. The heart beat of the fetus is visible in some signals – most clearly in the first one.

We next scale the components to have unit variances to make the elements of the unmixing matrix larger. Then the JADE estimate is computed and resulting components are plotted in Figure 5.

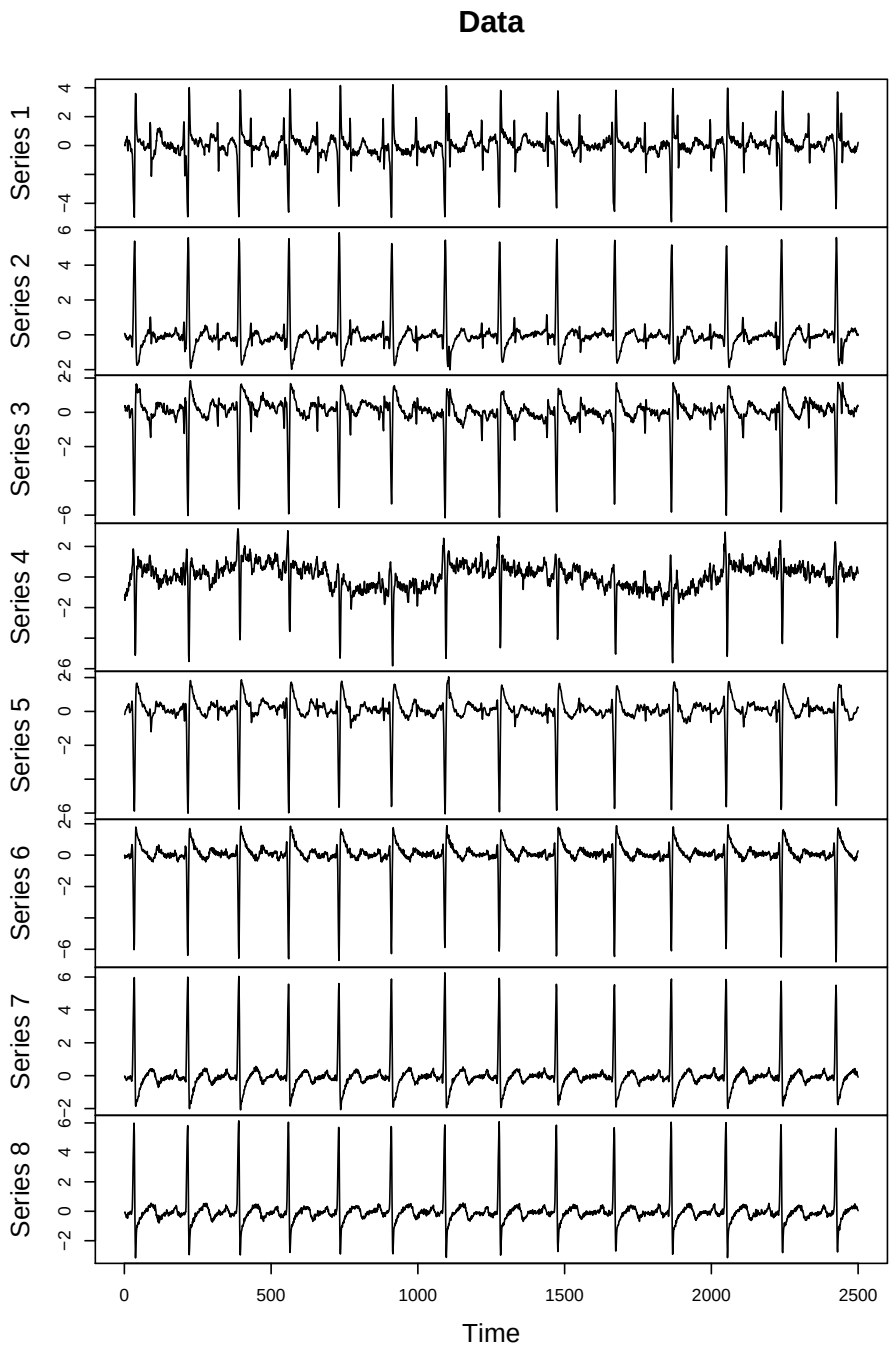


Figure 4: Electrocardiography recordings of a pregnant woman.

```
R> scale(X, center = FALSE, scale = apply(X, 2, sd))
R> jade <- JADE(X)
R> plot.ts(bss.components(jade), nc = 1, main = "JADE solution")
```

In Figure 5 it can be seen that the first three components are related to the mother's heartbeat and the fourth component is related to the fetus's heartbeat. Since we are interested in the

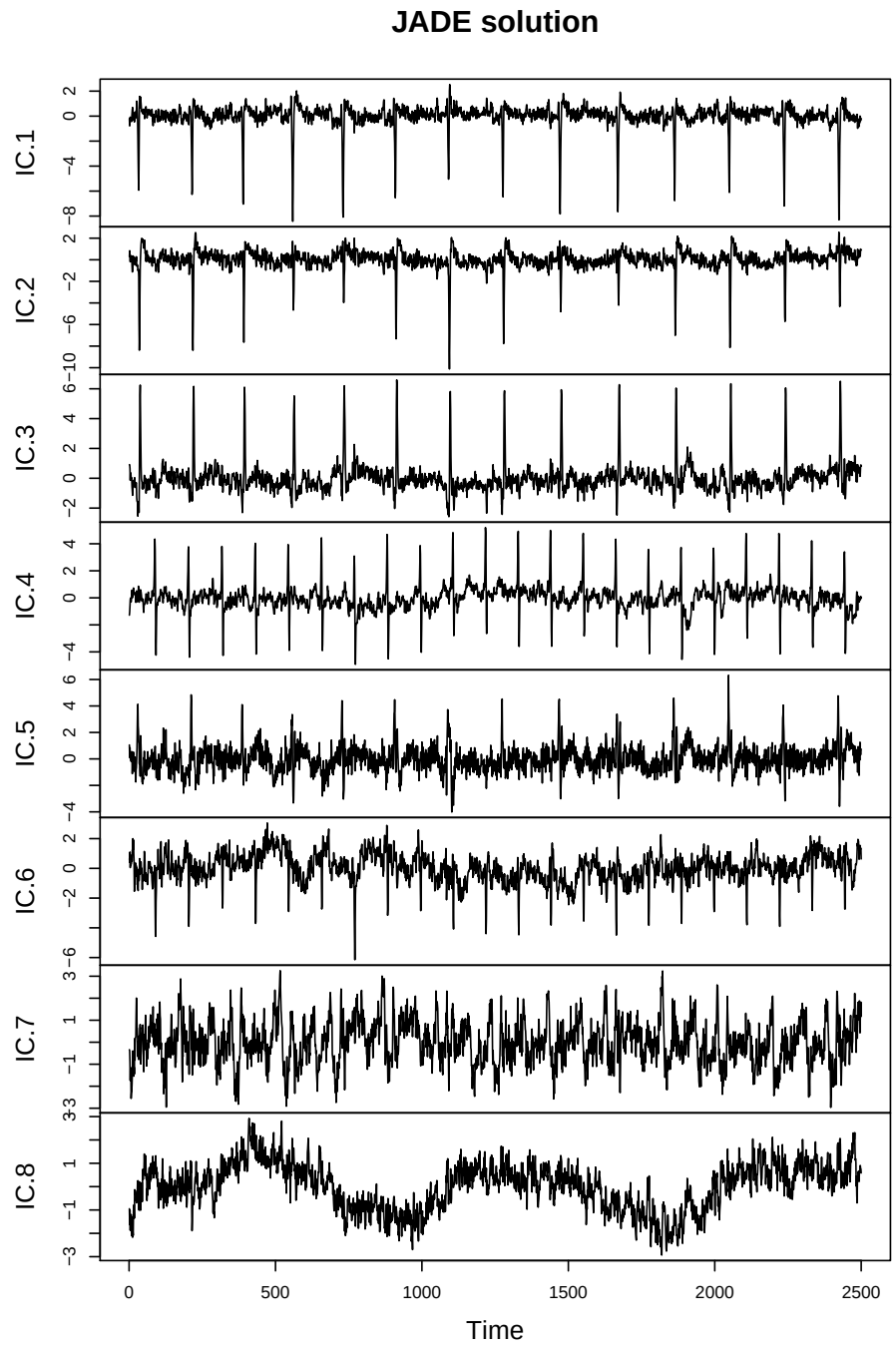


Figure 5: The independent components estimated with the JADE method.

fourth component, we pick up the corresponding coefficients from the fourth row of the unmixing matrix estimate. For demonstration purposes, we also derive their standard errors in order to see how much uncertainty is included in the results. These would be useful for example when selecting the best BSS method in a case where estimation accuracy of only one component is of interest, as opposed to Example 2 where the whole unmixing matrix was considered.

```
R> ascov <- ASCOV_JADE_est(X)
R> Vars <- matrix(diag(ascov$COV_W), nrow = 8)
R> Coefs <- coef(jade)[4, ]
R> SDs <- sqrt(Vars[4, ])
R> Coefs

[1]  0.58797  0.74451 -1.91649 -0.01493  3.35648 -0.26278  0.78499  0.18756

R> SDs

[1] 0.07210 0.15221 0.10519 0.03859 0.14785 0.09713 0.26431 0.17951
```

Furthermore, we can test, for example, whether the recordings from the mother's chest area contribute to the estimate of the fourth component (fetus's heartbeat), i.e., whether the last three elements of the fourth row of the unmixing are non-zero. Since the JADE estimate is asymptotically multivariate normal, we can compute the Wald test statistic related to the null hypothesis $H_0 : ((W)_{46}, (W)_{47}, (W)_{48}) = (0, 0, 0)$. Notice that `ascov$COV_W` is the covariance matrix estimate of the vector built from the columns of the unmixing matrix estimate. Therefore we create the vector `w` and hypothesis matrix `L` accordingly. The sixth, seventh and eighth element of the fourth row of the $8 \times 8$ matrix are the $5 \cdot 8 + 4 = 44$th, $6 \cdot 8 + 4 = 52$nd and $7 \cdot 8 + 4 = 60$th elements of `w`, respectively.

```
R> w <- as.vector(coef(jade))
R> V <- ascov$COV_W
R> L1 <- L2 <- L3 <- rep(0, 64)
R> L1[5 * 8 + 4] <- L2[6 * 8 + 4] <- L3[7 * 8 + 4] <- 1
R> L <- rbind(L1, L2, L3)
R> Lw <- L %*% w
R> T <- t(Lw) %*% solve(L %*% tcrossprod(V, L), Lw)
R> T

     [,1]
[1,] 89.8

R> format.pval(1 - pchisq(T, 3))

[1] "<2e-16"
```

The very small $p$ value suggests that not all of the three elements are zero.

## 9. Conclusions

In this paper we have introduced the R packages **JADE** and **BSSasymp** which contain several practical tools for blind source separation.

Package **JADE** provides methods for three common BSS models. The functions allow the user to perform blind source separation in cases where the source signals are (i) independent and

identically distributed, (ii) weakly stationary time series, or (iii) time series with nonstationary variance. All BSS methods included in the package utilize either simultaneous diagonalization of two matrices or approximate joint diagonalization of several matrices. In order to make the package self-contained we have included in it several algorithms for joint diagonalization. Two of the algorithms, deflation-based joint diagonalization and joint diagonalization using Givens rotations, are described in detail in this paper.

Package **BSSasymp** provides tools to compute the asymptotic covariance matrices as well as their data-based estimates for most of the BSS estimators included in the package **JADE**. The functions allow the user to study the uncertainty in the estimation either in simulation studies or in practical applications. Notice that package **BSSasymp** is the first R package so far to provide such variance estimation methods for practitioners.

We have provided four examples to introduce the functionality of the packages. The examples show in detail (i) how to compare different BSS methods using artificial example (cocktail-party problem) or simulated data, (ii) how to select a best method for the problem at hand, and (iii) how to perform blind source separation with real data (ECG recording).

## Acknowledgments

We wish to thank the reviewers, the associate editor and the editor Achim Zeileis for their helpful comments and advice. Furthermore would we like to express our sincere gratitude to Docent Ella Bingham and Prof. Lieven De Lathauwer for making the data used in this paper available. This research was supported by the Academy of Finland (grants 251965, 256291 and 268703).

## References

Amari SI, Cichocki A, Yang HH (1996). "A New Learning Algorithm for Blind Source Separation." In *Advances in Neural Information Processing Systems 8*, pp. 757–763. MIT Press, Cambridge.

Belouchrani A, Abed-Meraim K, Cardoso JF, Moulines E (1997). "A Blind Source Separation Technique Using Second-Order Statistics." *IEEE Transactions on Signal Processing*, **45**(2), 434–444. doi:10.1109/78.554307.

Cardoso JF (1989). "Source Separation Using Higher Order Moments." In *Proceedings of the IEEE International Conference on Accoustics, Speech and Signal Processing*, pp. 2109–2112. Glasgow.

Cardoso JF, Souloumiac A (1993). "Blind Beamforming for Non Gaussian Signals." *IEE Proceedings F (Radar and Signal Processing)*, **140**(6), 362–370. doi:10.1049/ip-f-2.1993.0054.

Choi S, Cichocki A (2000a). "Blind Separation of Nonstationary and Temporally Correlated Sources from Noisy Mixtures." In *Proceedings of the 2000 IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing X*, pp. 405–414. IEEE. doi:10.1109/NNSP.2000.889432.

Choi S, Cichocki A (2000b). "Blind Separation of Nonstationary Sources in Noisy Mixtures." *Electronics Letters*, **36**(9), 848–849. doi:10.1049/el:20000623.

Choi S, Cichocki A, Belouchrani A (2001). "Blind Separation of Second-Order Nonstationary and Temporally Colored Sources." In *Proceedings of the 11th IEEE Signal Processing Workshop on Statistical Signal Processing, 2001*, pp. 444–447. IEEE.

Cichocki A, Amari S (2002). *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications*. John Wiley & Sons, Cichester.

Cichocki A, Amari S, Siwek K, Tanaka T, Phan AH, others (2014). ***ICALAB*** *Toolboxes*. URL http://www.bsp.brain.riken.jp/ICALAB.

Clarkson DB (1988). "Remark AS R71: A Remark on Algorithm AS 211. The F-G Diagonalization Algorithm." *Journal of the Royal Statistical Society C*, **37**(1), 147–151. doi:10.2307/2347513.

Comon P, Jutten C (2010). *Handbook of Blind Source Separation. Independent Component Analysis and Applications*. Academic Press, Amsterdam.

De Lathauwer L, De Moor B, Vandewalle J (1995). "Fetal Electrocardiogram Extraction by Source Subspace Separation." In *IEEE SP/Athos Workshop on Higher-Order Statistics*, pp. 134–138. IEEE.

Flury B (1984). "Common Principal Components in $k$ Groups." *Journal of the American Statistical Association*, **79**(388), 892–898. doi:10.2307/2288721.

Golub GH, Van Loan CF (2002). *Matrix Computations*. Johns Hopkins University Press, Baltimore.

Gouy-Pailler C (2009). ***jointDiag****: Joint Approximate Diagonalization of a Set of Square Matrices*. R package version 0.2, URL https://CRAN.R-project.org/package=jointDiag.

Hastie T, Tibshirani R (2010). ***ProDenICA****: Product Density Estimation for ICA Using Tilted Gaussian Density Estimates*. R package version 1.0, URL https://CRAN.R-project.org/package=ProDenICA.

Helwig NE (2015). ***ica****: Independent Component Analysis*. R package version 1.0-1, URL https://CRAN.R-project.org/package=ica.

Hyvärinen A, Karhunen J, Oja E (2001). *Independent Component Analysis*. John Wiley & Sons, New York.

Ilmonen P, Nevalainen J, Oja H (2010a). "Characteristics of Multivariate Distributions and the Invariant Coordinate System." *Statistics and Probability Letters*, **80**(23–24), 1844–1853. doi:10.1016/j.spl.2010.08.010.

Ilmonen P, Nordhausen K, Oja H, Ollila E (2010b). "A New Performance Index for ICA: Properties, Computation and Asymptotic Analysis." In V Vigneron, V Zarzoso, E Moreau, R Gribonval, E Vincent (eds.), *Latent Variable Analysis and Signal Separation – 9th International Conference, LVA/ICA 2010, St. Malo, France, September 27–30, 2010. Proceedings*, volume 6365 of *Lecture Notes in Computer Science*, pp. 229–236. Springer-Verlag.

Ilmonen P, Nordhausen K, Oja H, Ollila E (2012). "On Asymptotics of ICA Estimators and Their Performance Indices." arXiv:1212.3953 [stat.ME], URL http://arxiv.org/abs/1212.3953.

Joyce CA, Gorodnitsky IF, Kutas M (2004). "Automatic Removal of Eye Movement and Blink Artifacts from EEG Data Using Blind Component Separation." *Psychophysiology*, **41**, 313–325.

Karvanen J (2009). ***PearsonICA**: Independent Component Analysis Using Score Functions from the Pearson System*. R package version 1.2-4, URL https://CRAN.R-project.org/package=PearsonICA.

Ligges U, Krey S, Mersmann O, Schnackenberg S (2016). ***tuneR**: Analysis of Music and Speech*. R package version 1.3.1, URL https://CRAN.R-project.org/package=tuneR.

Marchini JL, Heaton C, Ripley BD (2013). ***fastICA**: FastICA Algorithms to Perform ICA and Projection Pursuit*. R package version 1.2-0, URL https://CRAN.R-project.org/package=fastICA.

Miettinen J, Illner K, Nordhausen K, Oja H, Taskinen S, Theis FJ (2016). "Separation of Uncorrelated Stationary Time Series Using Autocovariance Matrices." *Journal of Time Series Analysis*, **37**(3), 337–354. doi:10.1111/jtsa.12159.

Miettinen J, Nordhausen K, Oja H, Taskinen S (2012). "Statistical Properties of a Blind Source Separation Estimator for Stationary Time Series." *Statistics and Probability Letters*, **82**(11), 1865–1873. doi:10.1016/j.spl.2012.06.025.

Miettinen J, Nordhausen K, Oja H, Taskinen S (2013). "Fast Equivariant JADE." In *Proceedings of 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013)*, pp. 6153–6157. IEEE. doi:10.1109/icassp.2013.6638847.

Miettinen J, Nordhausen K, Oja H, Taskinen S (2014). "Deflation-Based Separation of Uncorrelated Stationary Time Series." *Journal of Multivariate Analysis*, **123**, 214–227. doi:10.1016/j.jmva.2013.09.009.

Miettinen J, Nordhausen K, Oja H, Taskinen S (2015a). ***fICA**: Classic, Reloaded and Adaptive FastICA Algorithms*. R package version 1.0-3, URL https://CRAN.R-project.org/package=fICA.

Miettinen J, Nordhausen K, Oja H, Taskinen S (2017). ***BSSasymp**: Asymptotic Covariance Matrices of Some BSS Mixing and Unmixing Matrix Estimates*. R package version 1.2-0, URL https://CRAN.R-project.org/package=BSSasymp.

Miettinen J, Taskinen S, Nordhausen K, Oja H (2015b). "Fourth Moments and Independent Component Analysis." *Statistical Science*, **30**(3), 372–390. doi:10.1214/15-sts520.

Nordhausen K (2014). "On Robustifying Some Second Order Blind Source Separation Methods for Nonstationary Time Series." *Statistical Papers*, **55**(1), 141–156. doi:10.1007/s00362-012-0487-5.

Nordhausen K, Cardoso JF, Miettinen J, Oja H, Ollila E, Taskinen S (2017). ***JADE****: Blind Source Separation Methods Based on Joint Diagonalization and Some BSS Performance Criteria*. R package version 2.0-0, URL https://CRAN.R-project.org/package=JADE.

Nordhausen K, Gutch HW, Oja H, Theis FJ (2012). "Joint Diagonalization of Several Scatter Matrices for ICA." In FJ Theis, A Cichocki, A Yeredor, M Zibulevsky (eds.), *Latent Variable Analysis and Signal Separation – 10th International Conference, LVA/ICA 2012, Tel Aviv, Israel, March 12–15, 2012. Proceedings*, volume 7191 of *Lecture Notes in Computer Science*, pp. 172–179. Springer-Verlag.

Nordhausen K, Oja H, Ollila E (2008a). "Robust Independent Component Analysis Based on Two Scatter Matrices." *Austrian Journal of Statistics*, **37**(1), 91–100.

Nordhausen K, Oja H, Tyler DE (2008b). "Tools for Exploring Multivariate Data: The Package **ICS**." *Journal of Statistical Software*, **28**(6), 1–31. doi:10.18637/jss.v028.i06.

Nordhausen K, Ollila E, Oja H (2011). "On the Performance Indices of ICA and Blind Source Separation." In *Proceedings of 2011 IEEE 12th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC 2011)*, pp. 486–490. IEEE.

Nordhausen K, Tyler DE (2015). "A Cautionary Note on Robust Covariance Plug-In Methods." *Biometrika*, **102**(3), 573–588. doi:10.1093/biomet/asv022.

Oja H, Nordhausen K (2012). "Independent Component Analysis." In AH El-Shaarawi, W Piegorsch (eds.), *Encyclopedia of Environmetrics*, 2nd edition, pp. 1352–1360. John Wiley & Sons, Chichester.

Oja H, Sirkiä S, Eriksson J (2006). "Scatter Matrices and Independent Component Analysis." *Austrian Journal of Statistics*, **35**(2–3), 175–189.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Teschendorff A (2012). ***mlica2****: Independent Component Analysis Using Maximum Likelihood*. R package version 2.1, URL https://CRAN.R-project.org/package=mlica2.

The MathWorks Inc (2014). *MATLAB – The Language of Technical Computing, Version R2014b*. Natick, Massachusetts. URL http://www.mathworks.com/products/matlab/.

Tong L, Soon VC, Huang YF, Liu R (1990). "AMUSE: A New Blind Identification Algorithm." In *Proceedings of the IEEE International Symposium on Circuits and Systems 1990*, pp. 1784–1787. IEEE.

Yeredor A (2002). "Non-Orthogonal Joint Diagonalization in the Leastsquares Sense with Application in Blind Source Separation." *IEEE Transactions on Signal Processing*, **50**(7), 1545–1553. doi:10.1109/tsp.2002.1011195.

Yu X, Hu D, Xu J (2014). *Blind Source Separation – Theory and Practise*. John Wiley & Sons, Singapore. doi:10.1002/9781118679852.

**Affiliation:**

Jari Miettinen, Sara Taskinen
Department of Mathematics and Statistics
University of Jyvaskyla
40014 University of Jyvaskyla, Finland
E-mail: jari.p.miettinen@jyu.fi, sara.l.taskinen@jyu.fi

Klaus Nordhausen
Department of Mathematics and Statistics
University of Turku
20014 University of Turku, Finland
E-mail: klaus.nordhausen@utu.fi

*Journal of Statistical Software* http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
January 2017, Volume 76, Issue 2 *Submitted:* 2014-06-30
doi:10.18637/jss.v076.i02 *Accepted:* 2015-10-01