fICA: FastICA Algorithms and Their Improved Variants

by Jari Miettinen, Klaus Nordhausen, and Sara Taskinen

Abstract In independent component analysis (ICA) one searches for mutually independent non-gaussian latent variables when the components of the multivariate data are assumed to be linear combinations of them. Arguably, the most popular method to perform ICA is FastICA. There are two classical versions, the deflation-based FastICA where the components are found one by one, and the symmetric FastICA where the components are found simultaneously. These methods have been implemented previously in two R packages, **fastICA** and **ica**. We present the R package **fICA** and compare it to the other packages. Additional features in **fICA** include optimization of the extraction order in the deflation-based version, possibility to use any nonlinearity function, and improvement to convergence of the deflation-based algorithm. The usage of the package is demonstrated by applying it to the real ECG data of a pregnant woman.

Introduction

The goal of independent component analysis is to transform a multivariate dataset so that the resulting components are as independent as possible. The idea is to separate latent components from the dataset which is assumed to consist of linear mixtures of them.

The basic independent component (IC) model is thus written as

$$\boldsymbol{x}_i = A\boldsymbol{z}_i,\ i = 1,\ldots,n, \tag{1}$$

where $X = (\boldsymbol{x}_1,\ldots,\boldsymbol{x}_n)$ is a centered $p \times n$ data matrix, A is a $p \times p$ mixing matrix, and $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_n$ are realizations of a zero mean random p-vector $\boldsymbol{z} = (z_1,\ldots,z_p)^T$ with mutually independent components and at most one Gaussian component. The aim of ICA is then to estimate the unmixing matrix $W = A^{-1}$ and/or Z from the data matrix X alone (Hyvärinen et al., 2001; Cichocki and Amari, 2002; Comon and Jutten, 2010; Nordhausen and Oja, 2018). Indeed, ensuring independence and nongaussianity of the components of $\boldsymbol{z}$ is sufficient for their consistent estimation, but only up to signs and scales (Comon, 1994). The scales of the independent components are often fixed by the assumption $\mathrm{Cov}(\boldsymbol{z}) = I_p$, and we may proceed from Model 1 to

$$\boldsymbol{x}_{st} = U^T\boldsymbol{z}, \tag{2}$$

where $\boldsymbol{x}_{st} = \mathrm{Cov}(\boldsymbol{x})^{-1/2}(\boldsymbol{x} - \mathbb{E}(\boldsymbol{x}))$ and U is an orthogonal $p \times p$ matrix. This means the problem can be reduced to that of finding an orthogonal matrix and the final unmixing matrix estimate is then $W = U\mathrm{Cov}(\boldsymbol{x})^{-1/2}$.

ICA has become a widely used tool in various fields, including brain imaging, image and audio signal processing, and financial time series analysis. Despite the vast amount of literature on ICA methodology, the classical FastICA (Hyvärinen and Oja, 1997; Hyvärinen, 1999) methods are still the most popular tools with which to estimate the independent components. FastICA functions are available for several programming languages such as Matlab, in package FastICA (Gävert et al., 2005); R, in packages **fastICA** (Marchini et al., 2013) and **ica** (Helwig, 2015); C++, as part of package IT++ (Cayre and Furon, 2004); and Python, as part of packages MDP (Zito et al., 2008) and Scikit-learn (Pedregosa et al., 2011).

Here we introduce the R package **fICA** (Miettinen et al., 2017b). In addition to the classical FastICA methods, **fICA** includes functions for three recently proposed improved variants of FastICA, which are presented in the following section. Then we discuss how the **fICA** package differs from the packages **fastICA** and **ica**, and finally, we give some examples on the usage of the **fICA** package.

FastICA estimators in the fICA package

The **fICA** package includes implementations of the classical FastICA estimators as well as three improved variants, which we will now describe in detail. For other variants of FastICA, see Koldovský and Tichavský (2015).

All the methods maximize the nongaussianity of the components of $U\boldsymbol{x}_{st}$, when the nongaussianity of a univariate random variable x is measured by $|\,\mathbb{E}[G(x)]|$ with some twice continuously differentiable and nonquadratic function G which satisfies $\mathbb{E}[G(y)] = 0$ for the standard Gaussian

random variable y. The established and most popular options for G are

$$\begin{aligned}
&\textit{pow3}\text{: } G(x) = (x^4 - 3)/4,\\
&\textit{tanh}\text{: } G(x) = log(cosh(x)) - c_t\ ,\\
&\textit{gaus}\text{: } G(x) = -\exp(-x^2/2) - c_g\ ,
\end{aligned}$$

where $c_t = \mathbb{E}[\log(\cosh(y))] \approx 0.375$ and $c_g = \mathbb{E}[-\exp(-y^2/2)] \approx -0.707$. The names *pow3*, *tanh*, and *gaus* originate from the first derivatives of the functions, the so called nonlinearities, $g(x) = x^3$, $g(x) = \tanh(x)$, and $g(x) = x\exp(-x^2/2)$. The choice *pow3* always yields consistent estimates, whereas *tanh* and *gaus* do not. However, the examples (Miettinen et al., 2017c; Wei, 2014) of such distributions of the independent components where *tanh* and *gaus* fail are quite artificial, and we agree with Hyvärinen (1999) stating that *tanh* and *gaus* work with most of the reasonable distributions. For further discussions concerning possible nonlinearities and their properties, see Dermoune and Wei (2013) and Virta and Nordhausen (2017).

Deflation-based FastICA and its variants

The first FastICA paper (Hyvärinen and Oja, 1997) used *pow3* to find the independent components one after the other. In this *deflation-based FastICA*, the kth row of $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p)^T$ maximizes

$$|\,\mathbb{E}[G(\boldsymbol{u}_k^T \boldsymbol{x}_{st})]|$$

under the constraints $\boldsymbol{u}_k^T \boldsymbol{u}_k = 1$ and $\boldsymbol{u}_j^T \boldsymbol{u}_k = 0$ for $j = 1, \ldots, k-1$. A modified Newton-Raphson algorithm iterates the following steps until convergence:

$$\begin{aligned}
\boldsymbol{u}_k &\leftarrow \mathbb{E}[g(\boldsymbol{u}_k^T \boldsymbol{x}_{st})\boldsymbol{x}_{st}] - \mathbb{E}[g'(\boldsymbol{u}_k^T \boldsymbol{x}_{st})]\boldsymbol{u}_k\\
\boldsymbol{u}_k &\leftarrow \left(\boldsymbol{I}_p - \sum_{l=1}^{k-1} \boldsymbol{u}_l \boldsymbol{u}_l^T\right) \boldsymbol{u}_k\\
\boldsymbol{u}_k &\leftarrow ||\boldsymbol{u}_k||^{-1} \boldsymbol{u}_k
\end{aligned}$$

where the last two steps perform the Gram-Schmidt orthonormalization.

In addition to the global maximum, the objective function has several local maxima to which the FastICA algorithm may converge. The order in which the rows of U are found depends on their initial values. The extraction order affects the estimation performance, and to our knowledge, **fICA** is the only R package which provides a function to estimate the rows of U in such order that the objective function is maximized globally at each stage. This extraction order is usually better than a random order, even though not always optimal.

The statistical properties including asymptotic normality of the deflation-based FastICA estimator were studied rigorously in Ollila (2009, 2010); Nordhausen et al. (2011). The derivation of the asymptotic variances of the unmixing matrix estimate lead to *reloaded FastICA* (Nordhausen et al., 2011) which first computes an initial estimate $\widehat{Z}$ of the sources applying some simple IC method such as FOBI (Cardoso, 1989) or k-JADE (Miettinen et al., 2013). For all components of $\widehat{Z}$, one then computes certain quantities from which the optimal extraction order can be deduced.

The *adaptive deflation-based FastICA* (Miettinen et al., 2014) differs from the other estimators here, as instead of one nonlinearity it uses a candidate set of multiple nonlinearities, from which the best one is chosen for each component. Again, an initial estimate of the sources is computed and then the same quantities as in reloaded FastICA are obtained, but now for for each candidate in the set of nonlinearities. In addition to identifying the optimal extraction order, the optimal nonlinearity for each independent component is also chosen separately. The default set of nonlinearities for the function `adapt_fICA` in the R package **fICA** includes *pow3*, *tanh*, *gaus*, and the following eleven options:

$$\begin{array}{ll}
g(x) = x^3 & g(x) = (x-0.2)_+^2 + (x+0.2)_-^2\\
g(x) = tanh(x) & g(x) = (x-0.4)_+^2 + (x+0.4)_-^2\\
g(x) = xexp(-x^2/2) & g(x) = (x-0.6)_+^2 + (x+0.6)_-^2\\
g(x) = (x+0.6)_-^2 & g(x) = (x-0.8)_+^2 + (x+0.8)_-^2\\
g(x) = (x-0.6)_+^2 & g(x) = (x-1.2)_+^2 + (x+1.2)_-^2\\
g(x) = (x)_+^2 + (x)_-^2 & g(x) = (x-1.4)_+^2 + (x+1.4)_-^2\\
g(x) = (x-1)_+^2 + (x+1)_-^2 & g(x) = (x-1.6)_+^2 + (x+1.6)_-^2
\end{array}$$

where $(x)_+ = x$ if $x > 0$ and 0 otherwise, and $(x)_- = x$ if $x < 0$ and 0 otherwise.

Symmetric and squared symmetric FastICA

The *symmetric FastICA* (Hyvärinen, 1999) estimator maximizes

$$\sum_{j=1}^{p} |\,\mathbb{E}[G(\boldsymbol{u}_j^T \boldsymbol{x}_{st}]|$$

under the orthogonality constraint $UU^T = I_p$. Now the rows of $U = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_p)^T$ are estimated in parallel, and the steps of the iterative algorithm are

$$\begin{aligned}
&\boldsymbol{u}_k \leftarrow \mathbb{E}[g(\boldsymbol{u}_k^T \boldsymbol{x}_{st})\boldsymbol{x}_{st}] - \mathbb{E}[g'(\boldsymbol{u}_k^T \boldsymbol{x}_{st})]\boldsymbol{u}_k, \;\; k = 1, \ldots, p \\
&U \leftarrow (UU^T)^{-1/2} U.
\end{aligned}$$

The first update step is similar to that of the deflation-based version. In a way, the orthogonalization step can be seen as taking an average over the vectors of the first step, while in the Gram-Schmidt orthogonalization the errors in estimating the first rows of U will remain throughout the estimation. It is said that the error accumulates in the deflation-based approach. The symmetric FastICA is usually considered superior to the deflation-based FastICA, but there are also cases where the accumulation is preferable to the averaging. This occurs when some independent components are easier to find than the others. Statistical properties of symmetric FastICA are given in Miettinen et al. (2015); Wei (2015); Miettinen et al. (2017c).

The *squared symmetric FastICA* (Miettinen et al., 2017c) estimator maximizes

$$\sum_{j=1}^{p} (\mathbb{E}[G(\boldsymbol{u}_j^T \boldsymbol{x}_{st}])^2$$

under the orthogonality constraint $UU^T = I_p$. The first step of the algorithm

$$\begin{aligned}
&\boldsymbol{u}_k \leftarrow \mathbb{E}[G(\boldsymbol{u}_k^T \boldsymbol{x}_{st})](\mathbb{E}[g(\boldsymbol{u}_k^T \boldsymbol{x}_{st})\boldsymbol{x}_{st}] - \mathbb{E}[g'(\boldsymbol{u}_k^T \boldsymbol{x}_{st})]\boldsymbol{u}_k), \;\; k = 1, \ldots, p, \\
&U \leftarrow (UU^T)^{-1/2} U
\end{aligned}$$

is that of the classical symmetric version multiplied by $\mathbb{E}[G(\boldsymbol{u}_k^T \boldsymbol{x}_{st})]$. Hence, the squared symmetric version puts more weight on the rows which correspond to more nongaussian components, which most often, but not always, is advantageous.

For both of the symmetric versions, the initial value of U is not important when the sample size is large, as the algorithms converge always to the global maxima. With small sample sizes the initial value plays a role.

Comparison of the packages

The FastICA algorithms sometimes fail to converge when the sample size is small. However, this is not obvious if one uses the R package **fastICA** or **ica** since they do not give errors or warnings if the algorithm does not converge, but simply return the estimate after the maximum number of iterations has been reached. In the **fastICA** package, the information about convergence can be obtained, but since it is not given in the default setup, many of the users will not notice it. Similarly, in the **ica** package the number of iterations is included in the output and could be used as an indicator for convergence, but convergence problems might still go unnoticed. Growing the number of iterations does usually not help because if the algorithm has not converged after 2000 iterations, it is often repeating a loop of values. In contrast, the functions in the R package **fICA** give an error message if the algorithm does not converge. When computing the symmetric estimators, the user can choose the number of random orthogonal matrices which are generated for the initial values of the algorithms. If this number is at least two, the function returns the estimate which yields the highest value of the objective function and gives a warning if none of the algorithm runs converge. To avoid the repeating loop in the deflation-based case, **fICA** alternates the step size. The update of the vector $\boldsymbol{u}_k$ is of the form

$$\boldsymbol{u}_k \leftarrow w\boldsymbol{u}_k + (1 - w)\left(\mathbb{E}[g(\boldsymbol{u}_k^T \boldsymbol{x}_{st})\boldsymbol{x}_{st}] - \mathbb{E}[g'(\boldsymbol{u}_k^T \boldsymbol{x}_{st})]\boldsymbol{u}_k\right),$$

where w is zero for most iterations and takes small nonzero values with some selected iteration numbers with an irregular pattern.

To sum up the key differences of **fICA** to the **fastICA** and **ica** packages, we present Table 1.

The number of nonlinearities in **fICA** is basically unlimited. In addition to the already implemented suggestions, the user can provide their own functions by specifying g and its derivative. We provide

Table 1: The properties of the three R packages which include the FastICA algorithm.

	fICA	**fastICA**	**ica**
Symmetric FastICA	YES	YES	YES
Squared symm. FastICA	YES	NO	NO
Deflation-based FastICA	YES	YES	YES
Optimized extraction order	YES	NO	NO
Adaptive defl. FastICA	YES	NO	NO
Error if no convergence	YES	NO	NO
Number of nonlinearities	∞	2	3

an example of the user-specified option in the next section.

Examples

To illustrate the use of the **fICA** package, we give the following two examples.

Electrocardiography data

First, we analyze an electrocardiography recording (ECG) dataset `foetal_ecg.dat` (de Lathauwer et al., 2000), which can be downloaded from the supplementary files of Miettinen et al. (2017d). An ECG measures the electrical potential on the body surface. This recording is from a pregnant woman and consists of eight signals from five sensors on the stomach and three on the chest. The lengths of the signals are 2500 time points.

The signals are mixtures of fetus' and mother's heart beats and possibly some artifacts. The goal is to separate these sources using independent component analysis.

```
> library(fICA)
> library(JADE)
> options(digits = 4)
>
> dataset <- matrix(scan("https://www.jstatsoft.org/index.php/jss/article/
+ downloadSuppFile/v076i02/foetal_ecg.dat"), 2500, 9, byrow = TRUE)
> X <- dataset[ , 2:9]
> plot.ts(X, nc = 1, main = "Data")
```

After downloading the data and plotting it in Figure 1, we see that the main feature in each signal are 14 peaks, which are presumably produced by mother's heart beats. Next, we show how to apply the function `adapt_fICA` with a modified set of nonlinearities. We demonstrate how to use the pre-programmed *pow3* , *tanh*, and *gaus* options in the packages, and demonstrate how to use a user-specified function by providing g and its derivative for another classical nonlinearity called *skew* given by $g(x) = x^2$.

```
> g <- function(x){ x^2 }
> dg <- function(x){ 2*x }
> gf_new <- c(gf[1:3], g)
> dgf_new <- c(dgf[1:3], dg)
> gnames_new <- c(gnames[1:3], "skew")
>
> res <- adapt_fICA(X, gs = gf_new, dgs = dgf_new, name = gnames_new, kj = 1)
>
> res
W :
        [,1]      [,2]      [,3]      [,4]      [,5]       [,6]       [,7]
[1,] -0.04135  0.059521  0.004006  0.001754 -0.010123  0.0076023  0.0001551
[2,]  0.01731 -0.023294  0.002298  0.002663  0.008367  0.0022727 -0.0075911
[3,]  0.09352  0.071699 -0.106091 -0.004342  0.175369 -0.0030657 -0.0063677
[4,] -0.05358  0.106633  0.076979  0.044062 -0.024241 -0.0261067 -0.0522030
[5,]  0.15805  0.039095  0.233377 -0.041272 -0.169486  0.0104065 -0.0116267
[6,] -0.06605  0.114817  0.201392 -0.021387  0.234210 -0.0106505  0.0254791
[7,]  0.07817 -0.001272  0.146718  0.220175 -0.180189 -0.0008764  0.0035999
```

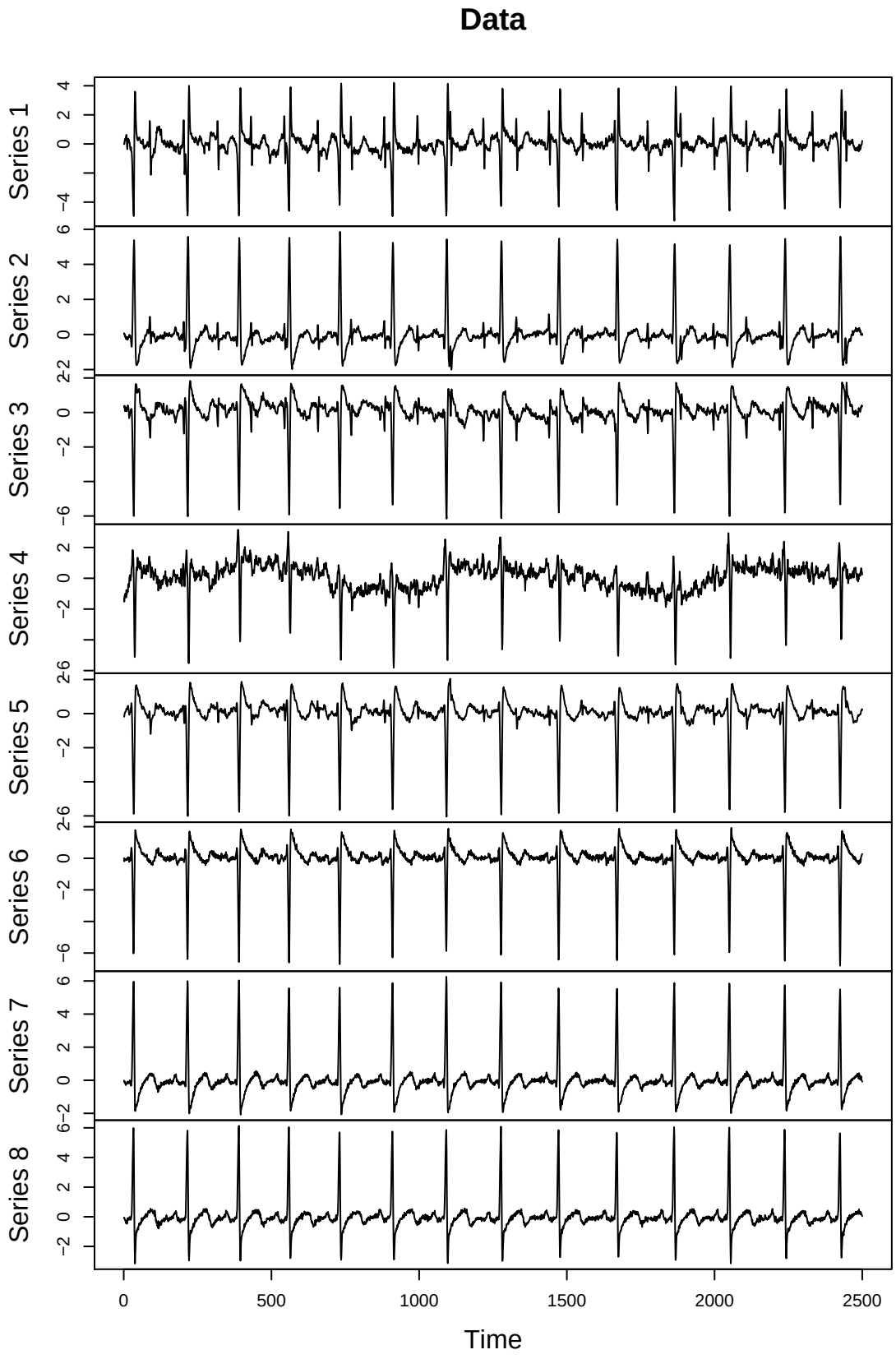

Figure 1: The ECG signals of a pregnant woman.

```
[8,] -0.25492  0.313232  0.049493  0.089799  0.028816  0.0232881 -0.0535985
          [,8]
[1,] -0.009234
[2,]  0.009301
[3,]  0.013034
[4,]  0.022396
[5,]  0.028662
[6,] -0.010261
[7,] -0.009922
[8,]  0.034743

gs :
[1] "pow3" "tanh" "gaus" "skew"

used_gs :
[1] "gaus" "gaus" "gaus" "tanh" "tanh" "tanh" "tanh"

alphas :
     comp 1 comp 2  comp 3 comp 4 comp 5 comp 6 comp 7  comp 8
pow3 0.7445 0.6119  1.4060  1.994  5.014  9.628  23.28 50087.8
tanh 0.2433 0.2569  0.7769  1.374  3.189  8.400  21.53   343.3
gaus 0.2058 0.2336  0.7336  1.458  3.351 10.829  22.87   215.1
skew 0.5457 0.4126 11.8980  7.091 12.867  9.118  25.00   108.7

init_est :
[1] "1-JADE"
```

In the output, we have the 8×8 unmixing matrix, the names of the available nonlinearities, the nonlinearities used, the criterion values for the selection of the extraction order and the nonlinearities, and the name of the initial estimator. Note that the `used_gs` output does not give a nonlinearity for the last component since the last component is fixed by the seven previous components due to the orthogonality constraint.

```
> plot.ts(bss.components(res), nc=1, main="Estimated components")
```

Figure 2 plots the estimated independent components. Clearly, the first two components are related to mother's heart beat and the third one is related to fetus' heart beat. The third row of the unmixing matrix estimate shows which data signals contribute to the estimate of the fetal heart beat. Not surprisingly, the nonzero coefficients correspond to sensors 1, 2, 3, and 5, which are located on the stomach area.

Simulations

In this simulation example, we also take a look at the related R package **BSSasymp** (Miettinen et al., 2017a), which computes asymptotic covariance matrices of several mixing and unmixing matrix estimates, including the FastICA variants discussed in this paper and implemented in **fICA**. The $p = 3$ independent components in the simulation setup are generated from a t_9-distribution, an exponential distribution with rate 1, and the normal distribution. Each density is standardized so that the expected value is 0 and the variance is 1. In each of the 1000 repetitions, the sample size is 5000, the mixing matrix is the same randomly generated 3×3 matrix, and we collect the unmixing matrix estimates given by (1) the reloaded deflation-based FastICA using *tanh* from **fICA**; (2) deflation-based FastICA using *tanh* from **fastICA**; and (3) deflation-based FastICA using *tanh* from **ica**. The goal of the simulation is to show that the extraction order of the components indeed matters.

```
> library(fICA)
> library(fastICA)
> library(ica)
> library(BSSasymp)
> library(JADE)
>
> options(digits = 4)
> set.seed(1145)
>
> rort <- function(p){qr.Q(qr(matrix(rnorm(p * p), p)))}
>
```

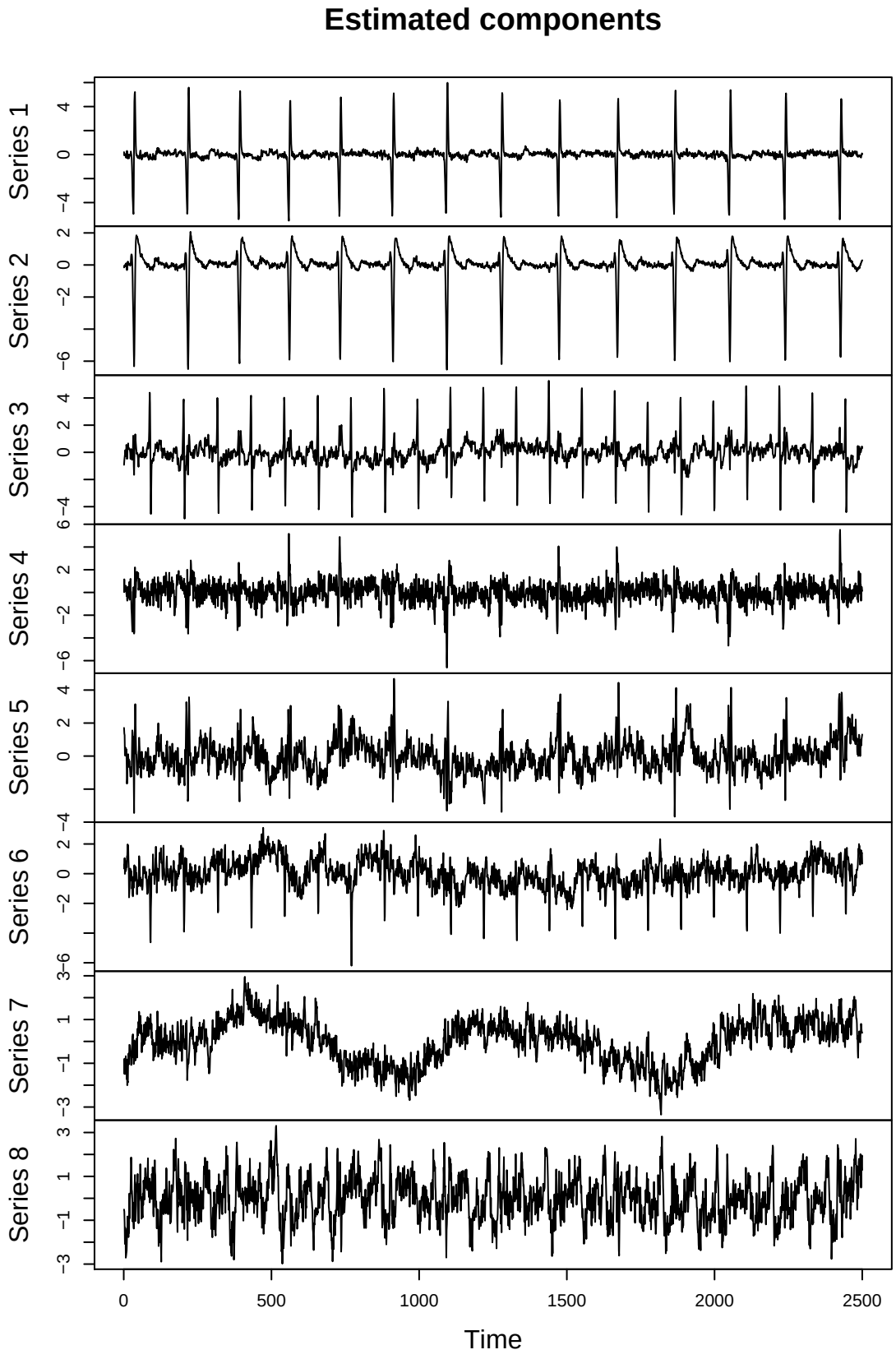

Figure 2: The estimated independent components from the ECG data.

```
>
> n <- 5000
> p <- 3
> A <- matrix(rnorm(p^2), p, p)
>
> Ws1 <- Ws2 <- Ws3 <- vector("list", 1000)
>
> for(i in 1:1000){
+   Z <- cbind(rt(n, 9) / sqrt(9/7), rexp(n, 1) - 1, rnorm(n))
+
+   X <- tcrossprod(Z,A)
+
+   init = rort(p)
+
+   Ws1[[i]] <- reloaded_fICA(X, g = "tanh")$W
+   res <- fastICA(X, n.comp = 3, alg.typ = "deflation" )
+   Ws2[[i]] <- t(res$W) %*% t(res$K)
+   Ws3[[i]] <- icafast(X, nc = 3, alg = "def", Rmat = init)$W
+ }
```

To obtain the theoretical asymptotic variances, the density functions of the independent components and their supports are required. Function `ASCOV_FastICAdefl` computes the values for deflation-based variants of FastICA. However, when the initial matrix is random as it is in **fastICA** and **ica**, there are no asymptotic results. Therefore, we only consider the reloaded FastICA for which the asymptotic variances are derived using `ASCOV_FastICAdefl` with `method = "adapt"` and only one nonlinearity. The output component `$COV_W` gives the asymptotic covariance matrix of the unmixing matrix estimate with the variances on the diagonal. The variances can be estimated using the function `ASCOV_FastICAdefl_est` and the data matrix. Here we estimate the variances from the last dataset generated above. In this case, the estimated variances are slightly higher than the theoretical values. Finally, we compute the variances from the simulations, and notice that they are quite close to the theoretical and the estimated variances.

```
> f1 <- function(x){ dt(x*sqrt(9/7),9)*sqrt(9/7) }
> f2 <- function(x){ dexp(x+1,1) }
> f3 <- function(x){ dnorm(x) }
> supp <- matrix(c(-Inf, -1, -Inf, Inf, Inf, Inf), p, 2)
>
> COV <- ASCOV_FastICAdefl(c(f1,f2,f3), gf[2], dgf[2], Gf[2],
+ method = "adapt", name = c("tanh"), supp = supp, A = A)$COV_W
> matrix(diag(COV), p, p)
      [,1]  [,2]   [,3]
[1,]  6.974 1.486  4.157
[2,] 27.813 5.163 10.895
[3,]  4.703 1.533  9.879
> COV_est <- ASCOV_FastICAdefl_est(X, gf[2], dgf[2], Gf[2],
+ method="adapt", name=c("tanh"))$COV_W
> matrix(diag(COV_est), p, p)*n
      [,1]  [,2]   [,3]
[1,]  7.219 1.369  4.356
[2,] 39.321 6.399 12.983
[3,]  5.714 1.552 16.177
> apply(simplify2array(Ws1), c(1,2), var)*n
      [,1]  [,2]   [,3]
[1,]  7.128 1.582  4.132
[2,] 29.009 5.334 11.498
[3,]  4.939 1.528 10.639
```

The minimum distance index (Ilmonen et al., 2010) can be used to quantify the separation performance of an unmixing matrix $\widehat{W}$ in a simulation study where the mixing matrix is known. It is defined as

$$\widehat{D} = \frac{1}{\sqrt{p-1}} \inf_{C \in \mathcal{C}} \|C\widehat{W}A - I_p\|,$$

where $\|\cdot\|$ is the Frobenius norm and

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

The minimum distance index takes values between 0 and 1, where zero means perfect separation. The function `ASCOV_FastICAdefl` gives the limiting expected value of $\widehat{D}^2 n(p-1)$, which is independent of the mixing matrix A. Below, the expected value is computed for the optimal extraction order (first the exponential, then the t_9-distribution, and the Gaussian component last), and for the other reasonable order (first the t_9-distribution, then exponential, and the Gaussian last). If the Gaussian component is found first or second, the limiting expected value goes to infinity. We also compute the averages of $\widehat{D}^2 n(p-1)$ over the 1000 repetitions for the three estimates. The average of the reloaded FastICA is close to the expected value of the optimal order, while the other two estimates with a random initial value have larger averages, indicating that the components are often found in non-optimal order.

```
> EMD_opt <- ASCOV_FastICAdefl(c(f1,f2,f3), gf[2], dgf[2], Gf[2],
+ method = "adapt", name = c("tanh"), supp = supp, A = A)$EMD
> EMD_opt
[1] 44.74
> EMD2 <- ASCOV_FastICAdefl(c(f1,f2,f3), gf[2], dgf[2], Gf[2],
+ method = "regular", name = c("tanh"), supp = supp, A = A)$EMD
> EMD2
[1] 67.66
> MD1 <- unlist(lapply(Ws1, MD, A=A))
> MD2 <- unlist(lapply(Ws2, MD, A=A))
> MD3 <- unlist(lapply(Ws3, MD, A=A))
>
> mean(MD1^2)*n*(p-1) # fICA
[1] 46.74
> mean(MD2^2)*n*(p-1) # fastICA
[1] 67.87
> mean(MD3^2)*n*(p-1) # ica
[1] 69.34
```

This simulation shows the effect of the extraction order in deflation-based fastICA. To date, only **fICA** has offered tools to find the optimal extraction order.

Summary

FastICA is the most widely used method to carry out independent component analysis. The R package **fICA** contributes to existing methods in two ways. First, it introduces the reloaded and adaptive deflation-based FastICA and the squared symmetric FastICA. Second, it improves on the existing R functions for classical FastICA algorithms by allowing user-specified nonlinearities and enhancing the convergence of the deflation-based FastICA. We gave a short review of the FastICA estimators, and examples on how to use them with the **fICA** package.

Acknowledgements

We would like to thank the anonymous reviewer for comments. The work of Klaus Nordhausen and Sara Taskinen was supported by CRoNoS COST Action IC1408, COST (European Cooperation in Science and Technology).

Bibliography

J. F. Cardoso. Source separation using higher order moments. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 2109–2112, 1989. doi: 10.1109/ICASSP.1989.266878. [p149]

F. Cayre and T. Furon. *Implementation of FastICA (Independent Component Analysis) for IT++*, 2004. TEMICS Project INRIA/Rennes (IRISA) Campus Universitaire de Beaulieu 35402. [p148]

A. Cichocki and S. Amari. *Adaptive Blind Signal and Image Processing: Learning Algorithms and Applications.* John Wiley & Sons, Cichester, 2002. doi: 10.1002/0470845899. [p148]

P. Comon. Independent component analysis - a new concept? *Signal Processing*, 36:287–314, 1994. doi: 10.1016/0165-1684(94)90029-9. [p148]

P. Comon and C. Jutten. *Handbook of Blind Source Separation. Independent Component Analysis and Applications*. Academic Press, Amsterdam, 2010. [p148]

L. de Lathauwer, B. de Moor, and J. Vandewalle. Fetal electrocardiogram extraction by blind source subspace separation. *IEEE Transactions on Biomedical Engineering*, 47(5):567–572, 2000. doi: 10.1109/10.841326. [p151]

A. Dermoune and T. Wei. FastICA algorithm: Five criteria for the optimal choice of the nonlinearity function. *IEEE Transactions on Signal Processing*, 61(8):2078–2087, 2013. doi: 10.1109/TSP.2013.2243440. [p149]

H. Gävert, J. Hurri, J. Särelä, and A. Hyvärinen. *The FastICA package for MATLAB*, 2005. Lab. of Computer and Information Science, Helsinki University of Technology. [p148]

N. E. Helwig. **ica***: Independent Component Analysis and Projection Pursuit*, 2015. URL http://CRAN.R-project.org/package=ica. R package version 1.0-1. [p148]

A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10:626–634, 1999. doi: 10.1109/72.761722. [p148, 149, 150]

A. Hyvärinen and E. Oja. A fast fixed-point algorithm for independent component analyis. *Neural Computation*, 9:1483–1492, 1997. doi: 10.1162/neco.1997.9.7.1483. [p148, 149]

A. Hyvärinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. John Wiley & Sons, New York, USA, 2001. [p148]

P. Ilmonen, K. Nordhausen, H. Oja, and E. Ollila. A new performance index for ICA: Properties computation and asymptotic analysis. In V. Vigneron, V. Zarzoso, E. Moreau, R. Gribonval, and E. Vincent, editors, *"Latent Variable Analysis and Signal Separation", LNCS*, volume 6365, pages 229–236, Heidelberg, 2010. Springer. doi: 10.1007/978-3-642-15995-4_29. [p155]

Z. Koldovský and P. Tichavský. Improved variants of the FastICA algorithm. *Advances in Independent Component Analysis and Learning Machines*, 53:53–74, 2015. doi: 10.1016/B978-0-12-802806-3.00002-6. [p148]

J. L. Marchini, C. Heaton, and B. D. Ripley. **fastICA***: FastICA Algorithms to Perform ICA and Projection Pursuit*, 2013. URL http://CRAN.R-project.org/package=fastICA. R package version 1.2-0. [p148]

J. Miettinen, K. Nordhausen, H. Oja, and S. Taskinen. Fast equivariant JADE. In *International Conference on Acoustics, Speech, and Signal Processing*, pages 6153–6157, 2013. doi: 10.1109/ICASSP.2013.6638847. [p149]

J. Miettinen, K. Nordhausen, H. Oja, and S. Taskinen. Deflation-based FastICA with adaptive choices of nonlinearities. *IEEE Transactions on Signal Processing*, 62:5716–5724, 2014. doi: 10.1109/TSP.2014.2356442. [p149]

J. Miettinen, S. Taskinen, K. Nordhausen, and H. Oja. Fourth moments and independent component analysis. *Statistical Science*, 30:372–390, 2015. doi: 10.1214/15-STS520. [p150]

J. Miettinen, K. Nordhausen, H. Oja, and S. Taskinen. **BSSasymp***: Asymptotic Covariance Matrices of Some BSS Mixing and Unmixing Matrix Estimates*, 2017a. URL http://CRAN.R-project.org/package=BSSasymp. R package version 1.2-1. [p153]

J. Miettinen, K. Nordhausen, H. Oja, and S. Taskinen. **fICA***: Classical, Reloaded and Adaptive FastICA Algorithms*, 2017b. URL http://CRAN.R-project.org/package=fICA. R package version 1.1-0. [p148]

J. Miettinen, K. Nordhausen, H. Oja, S. Taskinen, and J. Virta. The squared symmetric FastICA estimator. *Signal Processing*, 131:402–411, 2017c. doi: 10.1016/j.sigpro.2016.08.028. [p149, 150]

J. Miettinen, K. Nordhausen, and S. Taskinen. Blind source separation based on joint diagonalization in R: The packages **JADE** and **BSSasymp**. *Journal of Statistical Software*, 76:1–31, 2017d. doi: 10.18637/jss.v076.i02. [p151]

K. Nordhausen and H. Oja. Independent component analysis: A statistical perspective. *Wiley Interdisciplinary Reviews: Computational Statistics*, page e1440, 2018. doi: 10.1002/wics.1440. [p148]

K. Nordhausen, P. Ilmonen, A. Mandal, and H. Oja. Deflation-based FastICA reloaded. In *European Signal Processing Conference*, pages 1854–1858, 2011. [p149]

E. Ollila. On the robustness of the deflation-based FastICA estimator. In *IEEE Workshop on Statistical Signal Processing*, pages 673–676, 2009. doi: 10.1109/SSP.2009.5278485. [p149]

E. Ollila. The deflation-based FastICA estimator: statistical analysis revisited. *IEEE Transactions on Signal Processing*, 58:1527–1541, 2010. doi: 10.1109/TSP.2009.2036072. [p149]

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, ..., and J. Vanderplas. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2011, 2011. [p148]

J. Virta and K. Nordhausen. On the optimal non-linearities for Gaussian mixtures in FastICA. In P. Tichavský, M. Babaie-Zadeh, O. J. J. Michel, and N. Thirion-Moreau, editors, *Latent Variable Analysis and Signal Separation: 13th International Conference, LVA/ICA 2017, Grenoble, France, February 21-23, 2017, Proceedings*, pages 427–437, Cham, 2017. Springer International Publishing. doi: 10.1007/978-3-319-53547-0_40. [p149]

T. Wei. On the spurious solutions of the FastICA algorithm. In *IEEE Workshop on Statistical Signal Processing*, pages 161–164, 2014. doi: 10.1109/SSP.2014.6884600. [p149]

T. Wei. A convergence and asymptotic analysis of the generalized symmetric FastICA algorithm. *IEEE Transactions on Signal Processing*, 63(24):6445–6458, 2015. doi: 10.1109/TSP.2015.2468686. [p150]

T. Zito, N. Wilbert, L. Wiskott, and P. Berkes. Modular toolkit for data processing (mdp): a python data processing frame work. *Frontiers in Neuroinformatics*, 2, 2008. doi: 10.3389/neuro.11.008.2008. [p148]

Jari Miettinen
Department of Signal Processing and Acoustics
P.O.Box 15400, 20014 Aalto University
Finland
jari.p.miettinen@aalto.fi

Klaus Nordhausen
Institute of Statistics & Mathematical Methods in Economics
Vienna University of Technology Wiedner Hauptstr. 7, 1040 Vienna
Austria
klaus.nordhausen@tuwien.ac.at

Sara Taskinen
Department of Mathematics and Statistics
P.O.Box 35 (MaD), 40014 University of Jyväskylä
Finland
sara.l.taskinen@jyu.fi