BSS

$$\mathbf{X} = \mathbf{AS} + \mathbf{E}$$
$$\mathbf{x(t)} = \mathbf{As}(t) + \mathbf{e}(t)$$

$\mathbf{X}$	$(p \times N)$	rows = observed time series
$\mathbf{S}$	$(q \times N)$	rows = source time series (unobserved)
$\mathbf{A}$	$(p \times q)$	mixing matrix (to be estimated)
$\mathbf{E}$	$(p \times N)$	(spatially white) noise matrix

Goal: estimate $\mathbf{A}$ up to a permutation of its rows (essential equality)

- Allows us to estimate $\mathbf{S}$
- Note that columns of A reflect dynamic ranges of source series

Whitening

$$\mathbf{z(t)} = \mathbf{Us}(t) + \mathbf{We}(t)$$

$\mathbf{W}$	$(q \times p)$	whitening matrix
$\mathbf{U} = \mathbf{WA}$	$(q \times q)$	Unitary (complex orthogonal) whitened mixing matrix

Goal: estimate $\mathbf{U}$ from cov. matrix of whitened observations R_z, then solve for cov. matrix of sources R_s

$$R_z(\tau) = \mathbf{U}R_s(\tau)\mathbf{U}^H \qquad \tau \neq 0$$
$$R_s(\tau) = \mathbf{U}^H R_z(\tau)\mathbf{U} \qquad \tau \neq 0$$

Algorithm (SOBI)

1. Get sample covariance
2. Use this to estimate $\mathbf{W}$ and whitened signals $\mathbf{z}(t)$
3. Get sample estimates $\{\hat{R}_z(\tau_j)\}_{j=1}^J$ for some $J > 0$.
4. Estimate $\mathbf{U}$ as the joint diagonalizer of this set using JD criterion
5. Estimate $\mathbf{A}$ and $\mathbf{s}$ by solving in terms of $\hat{\mathbf{U}}$ and $\hat{\mathbf{W}}$

Cocktail Party Demo

Check out this documentation please

```
# --- ??JADE --------- #
library(JADE)
library(BSSasymp)
library(tuneR)

library(knitr)
library(kableExtra)
```

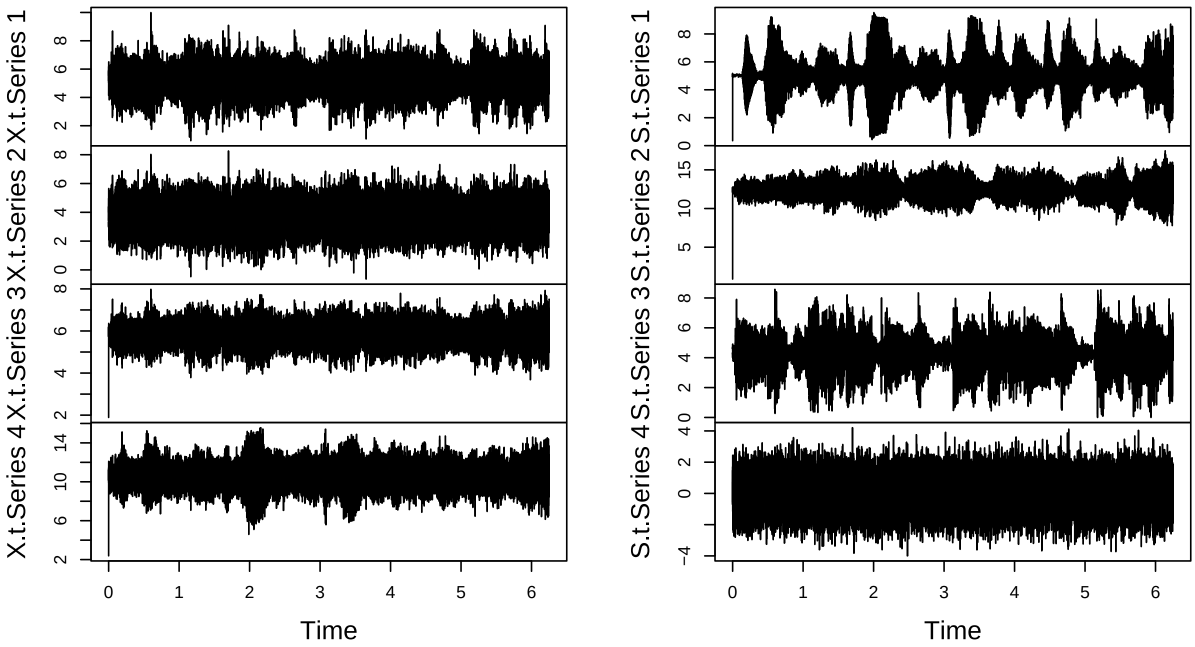
ICA: $x = zA^T + \mu$

(IC1) the source components are mutually independent,
(IC2) $E(z) = 0$ and $E(z^T z) = I_p$,
(IC3) at most one of the components is gaussian, and
(IC4) each source component is independent and identically distributed

```r
# --- Get source signals --------- #
S1 <- readWave(system.file("datafiles/source5.wav", package = "JADE"))
S2 <- readWave(system.file("datafiles/source7.wav", package = "JADE"))
S3 <- readWave(system.file("datafiles/source9.wav", package = "JADE"))
```

1. introduce noise component to the data
2. scale the components to have unit variances
3. generate components of mixing matrix from a std. normal distribution [except no? It says runif?]
4. mix the sources with mixing matrix

```r
set.seed(321)

N <- 50000 # series length
p <- 4     # number of observed series

# noise() outputs a formal wave object class, unlike rnorm()
NOISE <- noise("white", duration = 50000)

# get source matrix (transpose -- N x q)
S <- cbind(S1@left, S2@left, S3@left, NOISE@left)

# force each column of S to have unit variance
S <- scale(S, center = FALSE, scale = apply(S, 2, sd))

# format columns as time series, set total length = 6.25 seconds
S.t <- ts(S, start = 0, frequency = 8000)

# construct mixing matrix
A <- matrix(runif(p^2, 0, 1), p, p)

# mix sources with noise via mixing matrix
X <- tcrossprod(S.t, A) # this is S.t %*% t(A)

# Matrix of observed time series
X.t <- ts(X, start = 0, frequency = 8000)
```




```
# These are playable waves
x1 <- normalize(Wave(left = X[, 1], samp.rate = 8000, bit = 8), unit = "8")
x2 <- normalize(Wave(left = X[, 2], samp.rate = 8000, bit = 8), unit = "8")
x3 <- normalize(Wave(left = X[, 3], samp.rate = 8000, bit = 8), unit = "8")
x4 <- normalize(Wave(left = X[, 4], samp.rate = 8000, bit = 8), unit = "8")
```

JADE (**J**oint **A**pproximate **D**iagonalization of **E**igenmatrices)

J.-F. Cardoso and A. Souloumiac. Blind beamforming for non gaussian signals. In IEE Proceedings-F,volume 140, pages 362-370. IEEE, 1993.

SOBI (**S**econd **O**rder **B**lind **I**dentification)

THIS IS BELOU 97! In fact, this is the exact algorithm I outlined earlier in this document.

NSS-SD, NSS-JD and NSS-TD-JD (**N**onstationary **S**ource **S**eparation - considering multiple **T**ime **D**elayed correlation matrices - using **J**oint [or **S**imultaneous (old)] **D**iagonaliztion)

Choi and A. Cichocki. Blind separation of nonstationary sources in noisy mixtures. Electronics Letters, 36:848-849, 2000a.

```
# These are all estimates of the "unmixing matrix" A^-
jade <- JADE(X)    # subset of lags can be set using parameter 'k'
sobi <- SOBI(X.t)
nss.td.jd <- NSS.TD.JD(X.t)
```

To check, we want $\hat{A}^{-}A$ to have one unit entry per column/row, zeroes elsewhere, as per the "essential equality" described, below.

From Belouchrani97:
"Two matrices M and N are said to be essentially equal if there exists a matrix P such that $M = NP$, where P has exactly one nonzero entry in each row and column, where these entries have unit modulus."

The matrices $\hat{A}^{-}A$ are below, with minimum distance (MD) index

Table 1: JADE; MD = 0.075

-1.0000	0.0052	-0.0098	0.0041
-0.0050	0.0042	0.9994	-0.0389
0.0038	0.9964	-0.0091	-0.0826
0.0068	0.0843	0.0336	0.9958

Table 2: SOBI; MD = 0.0607

-0.0241	0.0075	0.9995	0.0026
-0.0690	0.9976	-0.0115	0.0004
-0.9973	-0.0683	-0.0283	-0.0025
0.0002	0.0009	-0.0074	1.0000

Table 3: NSS-TD-JD; MD = 0.0139

-0.0119	0.0045	0.9998	0.0016
-0.0009	0.0016	-0.0064	1.0000
0.0031	1.0000	-0.0065	-0.0001
0.9999	-0.0039	0.0170	0.0035

Selecting a set of Lags to improve SOBI

"The user needs to choose the value of T, the number of autocovariances to be used in the estimation. The value of T should be such that all lags with non-zero autocovariances are included, and the estimation of such autocovariances is still reliable. We choose T=1000." - From JADE documentation

```r
# Estimates (asymptotically) covariance matrix R_z
ascov1 <- ASCOV_SOBI_estN(X.t, taus = 1, M = 1000)
ascov2 <- ASCOV_SOBI_estN(X.t, taus = 1:3, M = 1000)
ascov3 <- ASCOV_SOBI_estN(X.t, taus = 1:12, M = 1000)
ascov4 <- ASCOV_SOBI_estN(X.t, taus = c(1, 2, 5, 10, 20), M = 1000)
ascov5 <- ASCOV_SOBI_estN(X.t, taus = 1:50, M = 1000)
ascov6 <- ASCOV_SOBI_estN(X.t, taus = c(1:20, (5:20) * 5), M = 1000)
ascov7 <- ASCOV_SOBI_estN(X.t, taus = 11:50, M = 1000)
```

```r
SumVar <- t(c(sum(diag(ascov1$COV_W)), sum(diag(ascov2$COV_W)),
              sum(diag(ascov3$COV_W)), sum(diag(ascov4$COV_W)),
              sum(diag(ascov5$COV_W)), sum(diag(ascov6$COV_W)),
              sum(diag(ascov7$COV_W))))

colnames(SumVar) <- c("(i)","(ii)","(iii)","(iv)","(v)","(vi)","(vii)")

MDs <- t(c(MD(ascov1$W,A), MD(ascov2$W,A),
```

```
            MD(ascov3$W,A), MD(ascov4$W,A),
            MD(ascov5$W,A), MD(ascov6$W,A), MD(ascov7$W,A)))
colnames(MDs) <- colnames(SumVar)
```

Table 4: Diagnostic: sum of limiting variances

(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)
363.035	0.1282037	0.1362057	0.0821712	0.075599	0.0679757	0.1268479

Table 5: Diagnostic: minimum distance index (only available because we constructed A)

(i)	(ii)	(iii)	(iv)	(v)	(vi)	(vii)
0.4329987	0.0365937	0.0607194	0.0337195	0.0124167	0.0123054	0.0120998

Thus we can use the sum of limiting variances approach (available through BSSasymp functions) in place of Minimum distance index when A is unknown.