

# Package ‘coroICA’

May 15, 2020

**Title** Confounding Robust Independent Component Analysis for Noisy and Grouped Data

**Version** 1.0.2

**Author** Niklas Pfister and Sebastian Weichwald

**Maintainer** Niklas Pfister <np@math.ku.dk>

**Description** Contains an implementation of a confounding robust independent component analysis (ICA) for noisy and grouped data. The main function `coroICA()` performs a blind source separation, by maximizing an independence across sources and allows to adjust for varying confounding based on user-specified groups. Additionally, the package contains the function `uwedge()` which can be used to approximately jointly diagonalize a list of matrices. For more details see the project website <<https://sweichwald.de/coroICA/>>.

**URL** <https://github.com/sweichwald/coroICA-R>

**BugReports** <https://github.com/sweichwald/coroICA-R/issues>

**Depends** R (>= 3.2.3)

**License** AGPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** stats, MASS

**RoxygenNote** 6.1.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2020-05-15 09:00:03 UTC

## R topics documented:

coroICA . . . . .	2
uwedge . . . . .	5
<b>Index</b>	<b>7</b>

---

 coroICA

 coroICA
 

---

### Description

Estimates the unmixing matrix  $V=A^{-1}$  of a confounded ICA model of the form  $X=AS+H$ , where  $H$  is confounding noise which is group-wise stationary and  $S$  are non-stationary signal sources. The function can also be used without a group-structure (i.e., using a single group) in which it corresponds to a noisy ICA that allows for arbitrary stationary noise  $H$ .

### Usage

```
coroICA(X, group_index = NA, partition_index = NA, n_components = NA,
        n_components_uwedge = NA, rank_components = FALSE,
        pairing = "complement", max_matrices = 1, groupsize = 1,
        partitionsize = NA, timelags = NA, instantcov = TRUE,
        max_iter = 1000, tol = 1e-12, minimize_loss = FALSE,
        condition_threshold = NA, silent = TRUE)
```

### Arguments

<code>X</code>	data matrix. Each column corresponds to one predictor variable.
<code>group_index</code>	vector coding to which group each sample belongs, with $\text{length}(\text{group\_index})=\text{nrow}(X)$ . If no group index is provided a rigid grid with <code>groupsize</code> samples per group is used (which defaults to all samples if <code>groupsize</code> was not set).
<code>partition_index</code>	vector coding to which partition each sample belongs, with $\text{length}(\text{partition\_index})=\text{nrow}(X)$ . If no partition index is provided a rigid grid with <code>partitionsize</code> samples per partition is used.
<code>n_components</code>	number of components to extract. If NA is passed, the same number of components as the input has dimensions is used.
<code>n_components_uwedge</code>	number of components to extract during uwedge approximate joint diagonalization of the matrices. If NA is passed, the same number of components as the input has dimensions is used.
<code>rank_components</code>	boolean, optional. When TRUE, the components will be ordered in decreasing stability.
<code>pairing</code>	either 'complement', 'neighbouring' or 'allpairs'. If 'allpairs' the difference matrices are computed for all pairs of partition covariance matrices, if 'complement' a one-vs-complement scheme is used and if 'neighbouring' differences with the right neighbour partition are used.
<code>max_matrices</code>	float or 'no_partitions', optional (default=1). The fraction of (lagged) covariance matrices to use during training or, if 'no_partitions', at most as many covariance matrices are used as there are partitions.

<code>groupsize</code>	int, optional. Approximate number of samples in each group when using a rigid grid as groups. If NA is passed, all samples will be in one group unless <code>group_index</code> is passed during fitting in which case the provided group index is used (the latter is the advised and preferred way).
<code>partitionsize</code>	int or vector of ints, optional. Approximate number of samples in each partition when using a rigid grid as partition. If NA is passed, a (hopefully sane) default is used, again, unless <code>partition_index</code> is passed during fitting in which case the provided partition index is used. If a vector is passed, each element is used to construct a grid and all resulting partitions are used.
<code>timelags</code>	vector of ints, optional. Specifies which timelags should be included. 0 corresponds to covariance matrix.
<code>instantcov</code>	boolean, default TRUE. Specifies whether to include covariance matrix when timelags are used.
<code>max_iter</code>	int, optional. Maximum number of iterations for the uwedge approximate joint diagonalisation during fitting.
<code>tol</code>	float, optional. Tolerance for terminating the uwedge approximate joint diagonalisation during fitting.
<code>minimize_loss</code>	boolean, optional. Parameter is passed to uwedge and specifies whether to compute loss function in each iteration step of uwedge.
<code>condition_threshold</code>	float, optional. Parameter is passed to uwedge and specifies whether and at which threshold to terminate uwedge iteration depending on the condition number of the unmixing matrix.
<code>silent</code>	boolean whether to suppress status outputs.

### Details

For further details see the references.

### Value

object of class 'CoroICA' consisting of the following elements

<code>V</code>	the unmixing matrix.
<code>covered</code>	boolean indicating whether the approximate joint diagonalisation converged due to <code>tol</code> .
<code>n_iter</code>	number of iterations of the approximate joint diagonalisation.
<code>meanoffdiag</code>	mean absolute value of the off-diagonal values of the to be jointly diagonalised matrices, i.e., a proxy of the approximate joint diagonalisation objective function.

### Author(s)

Niklas Pfister and Sebastian Weichwald

## References

Pfister, N., S. Weichwald, P. Bühlmann and B. Schölkopf (2018). Robustifying Independent Component Analysis by Adjusting for Group-Wise Stationary Noise ArXiv e-prints (arXiv:1806.01094).

Project website (<https://sweichwald.de/coroICA/>)

## See Also

The function `uwedge` allows to perform to perform an approximate joint matrix diagonalization.

## Examples

```
## Example
set.seed(1)

# Generate data from a block-wise variance model
d <- 2
m <- 10
n <- 5000
group_index <- rep(c(1,2), each=n)
partition_index <- rep(rep(1:m, each=n/m), 2)
S <- matrix(NA, 2*n, d)
H <- matrix(NA, 2*n, d)
for(i in unique(group_index)){
  varH <- abs(rnorm(d))/4
  H[group_index==i, ] <- matrix(rnorm(d*n)*rep(varH, each=n), n, d)
  for(j in unique(partition_index[group_index==i])){
    varS <- abs(rnorm(d))
    index <- partition_index==j & group_index==i
    S[index,] <- matrix(rnorm(d*n/m)*rep(varS, each=n/m),
                      n/m, d)
  }
}
A <- matrix(rnorm(d^2), d, d)
A <- A%%t(A)
X <- t(A%%t(S+H))

# Apply coroICA
res <- coroICA(X, group_index, partition_index, pairing="allpairs", rank_components=TRUE)

# Compare results
par(mfrow=c(2,2))
plot((S+H)[,1], type="l", main="true source 1", ylab="S+H")
plot(res$Shat[,1], type="l", main="estimated source 1", ylab="Shat")
plot((S+H)[,2], type="l", main="true source 2", ylab="S+H")
plot(res$Shat[,2], type="l", main="estimated source 2", ylab="Shat")
cor(res$Shat, S+H)
```

---

uwedge	<i>uwedge</i>
--------	---------------

---

### Description

Performs an approximate joint matrix diagonalization on a list of matrices. More precisely, for a list of matrices  $R_x$  the algorithm finds a matrix  $V$  such that for all  $i$   $V^T R_x[i] V$  is approximately diagonal.

### Usage

```
uwedge(Rx, init = NA, Rx0 = NA, return_diag = FALSE, tol = 1e-10,
       max_iter = 1000, n_components = NA, minimize_loss = FALSE,
       condition_threshold = NA, silent = TRUE)
```

### Arguments

<code>Rx</code>	list of matrices to be diagonalized.
<code>init</code>	matrix used in first step of initialization. If NA a default based on PCA is used
<code>Rx0</code>	matrix used for initial scaling.
<code>return_diag</code>	boolean. Specifies whether to return the list of diagonalized matrices.
<code>tol</code>	float, optional. Tolerance for terminating the iteration.
<code>max_iter</code>	int, optional. Maximum number of iterations.
<code>n_components</code>	number of components to extract. If NA is passed, all components are used.
<code>minimize_loss</code>	boolean whether to compute loss function in each iteration step and output $V$ with smallest loss over all iterations. Defaults to FALSE since it is computationally more expensive.
<code>condition_threshold</code>	float, optional. Stops iteration if condition number of $V$ passes this threshold. Default NA, means no threshold is used.
<code>silent</code>	boolean whether to suppress status outputs.

### Details

For further details see the references.

### Value

object of class 'uwedge' consisting of the following elements

<code>V</code>	joint diagonalizing matrix.
<code>Rxdiag</code>	list of diagonalized matrices.
<code>converged</code>	boolean specifying whether the algorithm converged for the given <code>tol</code> .
<code>iterations</code>	number of iterations of the approximate joint diagonalisation.

`meanoffdiag` mean absolute value of the off-diagonal values of the to be jointly diagonalised matrices, i.e., a proxy of the approximate joint diagonalisation objective function.

### Author(s)

Niklas Pfister and Sebastian Weichwald

### References

Pfister, N., S. Weichwald, P. Bühlmann and B. Schölkopf (2018). Robustifying Independent Component Analysis by Adjusting for Group-Wise Stationary Noise ArXiv e-prints (arXiv:1806.01094).  
Tichavsky, P. and Yeredor, A. (2009). Fast Approximate Joint Diagonalization Incorporating Weight Matrices. IEEE Transactions on Signal Processing.

### See Also

The function [coroICA](#) uses `uwedge`.

### Examples

```
## Example
set.seed(1)

# Generate data 20 matrix that can be jointly diagonalized
d <- 10
A <- matrix(rnorm(d*d), d, d)
A <- A%*%t(A)
Rx <- lapply(1:20, function(x) A %*% diag(rnorm(d)) %*% t(A))

# Perform approximate joint diagonalization
ptm <- proc.time()
res <- uwedge(Rx,
              return_diag=TRUE,
              max_iter=1000)
print(proc.time()-ptm)

# Average value of offdiagonal elements:
print(res$meanoffdiag)
```

# Index

coroICA, [2](#), [6](#)

uwedge, [4](#), [5](#)