Package ‘coroICA’

October 12, 2022

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

Index 7
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

```r
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

- **X**: data matrix. Each column corresponds to one predictor variable.
- **group_index**: vector coding to which group each sample belongs, with length(group_index) = nrow(X). If no group index is provided a rigid grid with groupsize samples per group is used (which defaults to all samples if groupsize was not set).
- **partition_index**: vector coding to which partition each sample belongs, with length(partition_index) = nrow(X). If no partition index is provided a rigid grid with partitionsize samples per partition is used.
- **n_components**: number of components to extract. If NA is passed, the same number of components as the input has dimensions is used.
- **n_components_uwedge**: 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.
- **rank_components**: boolean, optional. When TRUE, the components will be ordered in decreasing stability.
- **pairing**: 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.
- **max_matrices**: 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.
groupsize 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
group_index is passed during fitting in which case the provided group index is
used (the latter is the advised and preferred way).

partitionsize 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 partition_index 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.

timelags vector of ints, optional. Specifies which timelags should be included. 0 corre-
spends to covariance matrix.

instantcov boolean, default TRUE. Specifies whether to include covariance matrix when
timelags are used.

max_iter int, optional. Maximum number of iterations for the uwedge approximate joint
diagonalisation during fitting.

tol float, optional. Tolerance for terminating the uwedge approximate joint dia-
gonalisation during fitting.

minimize_loss boolean, optional. Parameter is passed to uwedge and specifies whether to com-
pute loss function in each iteration step of uwedge.

condition_threshold float, optional. Parameter is passed to uwedge and specifies whether and at
which threshold to terminate uwedge iteration depending on the condition num-
ber of the unmixing matrix.

silent boolean whether to supress status outputs.

Details

For further details see the references.

Value

object of class 'CoroICA' consisting of the following elements

V the unmixing matrix.

coverged boolean indicating whether the approximate joint diagonalisation converged due
to tol.

n_iter 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 func-
tion.

Author(s)

Niklas Pfister and Sebastian Weichwald
References


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

See Also

The function \texttt{uwedge} allows to perform to perform an approximate joint matrix diagonalization.

Examples

```r
## 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)
```
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 $i$ $V R_x[i] t(V)$ is approximately diagonal.

Usage

uwedge($R_x$, init = NA, $R_x0$ = NA, return_diag = FALSE, tol = $1e^{-10}$, max_iter = 1000, n_components = NA, minimize_loss = FALSE, condition_threshold = NA, silent = TRUE)

Arguments

- $Rx$: list of matrices to be diagonalized.
- init: matrix used in first step of initialization. If NA a default based on PCA is used.
- $R_x0$: matrix used for initial scaling.
- return_diag: boolean. Specifies whether to return the list of diagonalized matrices.
- tol: float, optional. Tolerance for terminating the iteration.
- max_iter: int, optional. Maximum number of iterations.
- n_components: number of components to extract. If NA is passed, all components are used.
- minimize_loss: 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.
- condition_threshold: float, optional. Stops iteration if condition number of $V$ passes this threshold. Default NA, means no threshold is used.
- silent: boolean whether to suppress status outputs.

Details

For further details see the references.

Value

object of class 'uwedge' consisting of the following elements

- $V$: joint diagonalizing matrix.
- $R_x$diag: list of diagonalized matrices.
- converged: boolean specifying whether the algorithm converged for the given tol.
- iterations: 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

See Also
The function coroICA uses uwedge.

Examples
```r
## 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