Package 'ica'

May 24, 2018

Type Package

Title Independent Component Analysis

Version 1.0-2

Date 2018-05-24

Author Nathaniel E. Helwig <helwig@umn.edu>

Maintainer Nathaniel E. Helwig <helwig@umn.edu>

Description Independent Component Analysis (ICA) using various algorithms: FastICA, Information-Maximization (Infomax), and Joint Approximate Diagonalization of Eigenmatrices (JADE).

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2018-05-24 06:12:53 UTC

R topics documented:

`ica-package` *Independent Component Analysis*

Description

Independent Component Analysis (ICA) using various algorithms: FastICA, Information-Maximization (Infomax), and Joint Approximate Diagonalization of Eigenmatrices (JADE).

Details

The DESCRIPTION file:

Package:	ica
Type:	Package
Title:	Independent Component Analysis
Version:	1.0-2
Date:	2018-05-24
Author:	Nathaniel E. Helwig <helwig@umn.edu>
Maintainer:	Nathaniel E. Helwig <helwig@umn.edu>
Description:	Independent Component Analysis (ICA) using various algorithms: FastICA, Information-Maximization (Infom
License:	GPL (>=2)

Index of help topics:

```
acy                     Amari-Cichocki-Yang Error
ica-package             Independent Component Analysis
icafast                 ICA via FastICA Algorithm
icaimax                 ICA via Infomax Algorithm
icajade                 ICA via JADE Algorithm
icaplot                 Plot Densities of Source Signal Distributions
icasamp                 Sample from Various Source Signal Distributions
```

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

Maintainer: Nathaniel E. Helwig <helwig@umn.edu>

References

Amari, S., Cichocki, A., & Yang, H.H. (1996). A new learning algorithm for blind signal separation. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems, 8*. Cambridge, MA: MIT Press.

Bach, F.R. (2002). *kernel-ica*. MATLAB toolbox (ver 1.2) http://www.di.ens.fr/~fbach/kernel-ica/.

Bach, F.R. & Jordan, M.I. (2002). Kernel independent component analysis. *Journal of Machine Learning Research, 3*, 1-48.

Bell, A.J. & Sejnowski, T.J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation, 7*, 1129-1159.

Cardoso, J.F., & Souloumiac, A. (1993). Blind beamforming for non-Gaussian signals. *IEE Proceedings-F, 140*, 362-370.

Cardoso, J.F., & Souloumiac, A. (1996). Jacobi angles for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications, 17*, 161-164.

Helwig, N.E. & Hong, S. (2013). A critique of Tensor Probabilistic Independent Component Analysis: Implications and recommendations for multi-subject fMRI data analysis. *Journal of Neuroscience Methods, 213*, 263-273.

Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks, 10*, 626-634.

Tucker, L.R. (1951). *A method for synthesis of factor analysis studies* (Personnel Research Section Report No. 984). Washington, DC: Department of the Army.

Examples

```
# See examples for icafast, icaimax, icajade, and icasamp
```

---

acy *Amari-Cichocki-Yang Error*

Description

The Amari-Cichocki-Yang (ACY) error is an asymmetric measure of dissimilarity between two nonsingular matrices `X` and `Y`. The ACY error: (a) is invariant to permutation and rescaling of the columns of `X` and `Y`, (b) ranges between 0 and `n-1`, and (c) equals 0 if and only if `X` and `Y` are identical up to column permutations and rescalings.

Usage

```
acy(X,Y)
```

Arguments

`X`	Nonsingular matrix of dimension $n \times n$ (test matrix).
`Y`	Nonsingular matrix of dimension $n \times n$ (target matrix).

Details

The ACY error is defined as

$$\frac{1}{2n}\sum_{i=1}^{n}\left(\frac{\sum_{j=1}^{n}|a_{ij}|}{\max_j |a_{ij}|}-1\right)+\frac{1}{2n}\sum_{j=1}^{n}\left(\frac{\sum_{i=1}^{n}|a_{ij}|}{\max_i |a_{ij}|}-1\right)$$

where $a_{ij} = (\mathbf{Y}^{-1}\mathbf{X})_{ij}$.

Value

Returns a scalar (the ACY error).

Warnings

If Y is singular, function will produce an error.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Amari, S., Cichocki, A., & Yang, H.H. (1996). A new learning algorithm for blind signal separation. In D. S. Touretzky, M. C. Mozer, and M. E. Hasselmo (Eds.), *Advances in Neural Information Processing Systems, 8*. Cambridge, MA: MIT Press.

Examples

```
##########   EXAMPLE   ##########

set.seed(1)
X <- matrix(runif(16),4,4)
Y <- matrix(runif(16),4,4)
Z <- X[,c(3,1,2,4)]%*%diag(1:4)
acy(X,Y)
acy(X,Z)
```

---

icafast — *ICA via FastICA Algorithm*

Description

Computes ICA decomposition using Hyvarinen's (1999) FastICA algorithm with various options.

Usage

```
icafast(X, nc, center = TRUE, maxit = 100, tol = 1e-6,
        Rmat = diag(nc), alg = c("par", "def"),
        fun = c("logcosh", "exp", "kur"), alpha = 1)
```

Arguments

`X`	Data matrix with `n` rows (samples) and `p` columns (variables).
`nc`	Number of components to extract.
`center`	If `TRUE`, columns of `X` are mean-centered before ICA decomposition.
`maxit`	Maximum number of algorithm iterations to allow.
`tol`	Convergence tolerance.
`Rmat`	Initial estimate of the `nc`-by-`nc` orthogonal rotation matrix.
`alg`	Algorithm to use: `alg="par"` to estimate all `nc` components in parallel (default) or `alg="def"` for deflation estimation (i.e., projection pursuit).
`fun`	Contrast function to use for negentropy approximation.
`alpha`	Tuning parameter for "logcosh" contrast function (1 <= `alpha` <= 2).

Details

ICA Model The ICA model can be written as `X=tcrossprod(S,M)+E`, where columns of `S` contain the source signals, `M` is the mixing matrix, and columns of `E` contain the noise signals. Columns of `X` are assumed to have zero mean. The goal is to find the unmixing matrix `W` such that columns of `S=tcrossprod(X,W)` are independent as possible.

Whitening Without loss of generality, we can write `M=P%*%R` where `P` is a tall matrix and `R` is an orthogonal rotation matrix. Letting `Q` denote the pseudoinverse of `P`, we can whiten the data using `Y=tcrossprod(X,Q)`. The goal is to find the orthongal rotation matrix `R` such that the source signal estimates `S=Y%*%R` are as independent as possible. Note that `W=crossprod(R,Q)`.

FastICA The FastICA algorithm finds the orthogonal rotation matrix `R` that (approximately) maximizes the negentropy of the estimated source signals. Negentropy is approximated using

$$J(s) = [E\{G(s)\} - E\{G(z)\}]^2$$

where *E* denotes the expectation, *G* is the contrast function, and *z* is a standard normal variable. See Hyvarinen (1999) for specifics of fixed-point algorithm.

Value

`S`	Matrix of source signal estimates (`S=Y%*%R`).
`M`	Estimated mixing matrix.
`W`	Estimated unmixing matrix (`W=crossprod(R,Q)`).
`Y`	Whitened data matrix.
`Q`	Whitening matrix.
`R`	Orthogonal rotation matrix.
`vafs`	Variance-accounted-for by each component.
`iter`	Number of algorithm iterations.
`alg`	Algorithm used (same as input).
`fun`	Contrast function (same as input).
`alpha`	Tuning parameter (same as input).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Helwig, N.E. & Hong, S. (2013). A critique of Tensor Probabilistic Independent Component Analysis: Implications and recommendations for multi-subject fMRI data analysis. *Journal of Neuroscience Methods, 213*, 263-273.

Hyvarinen, A. (1999). Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks, 10*, 626-634.

Examples

```
##########   EXAMPLE 1   ##########

# generate noiseless data (p==r)
set.seed(123)
nobs <- 1000
Amat <- cbind(icasamp("a","rnd",nobs),icasamp("b","rnd",nobs))
Bmat <- matrix(2*runif(4),2,2)
Xmat <- tcrossprod(Amat,Bmat)

# ICA via FastICA with 2 components
imod <- icafast(Xmat,2)
acy(Bmat,imod$M)
cor(Amat,imod$S)



##########   EXAMPLE 2   ##########

# generate noiseless data (p!=r)
set.seed(123)
nobs <- 1000
Amat <- cbind(icasamp("a","rnd",nobs),icasamp("b","rnd",nobs))
Bmat <- matrix(2*runif(200),100,2)
Xmat <- tcrossprod(Amat,Bmat)

# ICA via FastICA with 2 components
imod <- icafast(Xmat,2)
cor(Amat,imod$S)



##########   EXAMPLE 3   ##########

# generate noisy data (p!=r)
set.seed(123)
nobs <- 1000
Amat <- cbind(icasamp("a","rnd",nobs),icasamp("b","rnd",nobs))
Bmat <- matrix(2*runif(200),100,2)
```

```
Emat <- matrix(rnorm(10^5),1000,100)
Xmat <- tcrossprod(Amat,Bmat)+Emat

# ICA via FastICA with 2 components
imod <- icafast(Xmat,2)
cor(Amat,imod$S)
```

---

icaimax *ICA via Infomax Algorithm*

Description

Computes ICA decomposition using Bell and Sejnowski's (1995) Information-Maximization (Infomax) approach with various options.

Usage

```
icaimax(X, nc, center = TRUE, maxit = 100, tol = 1e-6,
        Rmat = diag(nc), alg = c("newton", "gradient"),
        fun = c("tanh", "log", "ext"), signs = rep(1, nc),
        signswitch = TRUE, rate = 1, rateanneal = NULL)
```

Arguments

`X`	Data matrix with `n` rows (samples) and `p` columns (variables).
`nc`	Number of components to extract.
`center`	If `TRUE`, columns of `X` are mean-centered before ICA decomposition.
`maxit`	Maximum number of algorithm iterations to allow.
`tol`	Convergence tolerance.
`Rmat`	Initial estimate of the `nc`-by-`nc` orthogonal rotation matrix.
`alg`	Algorithm to use: `alg="newton"` for Newton iteration, and `alg="gradient"` for gradient descent.
`fun`	Nonlinear (squashing) function to use for algorithm: `fun="tanh"` for hyperbolic tangent, `fun="log"` for logistic, and `fun="ext"` for extended Infomax.
`signs`	Vector of length `nc` such that `signs[j]==1` if j-th component is super-Gaussian and `signs[j]==-1` if j-th component is sub-Gaussian. Only used if `fun="ext"`. Ignored if `signswitch=TRUE`.
`signswitch`	If `TRUE`, the `signs` vector is automatically determined from the data; otherwise a confirmatory ICA decomposition is calculated using input `signs` vector. Only used if `fun="ext"`.
`rate`	Learing rate for gradient descent algorithm. Ignored if `alg="newton"`.
`rateanneal`	Annealing angle and proportion for gradient descent learing rate (see Details). Ignored if `alg="newton"`.

Details

ICA Model The ICA model can be written as X=tcrossprod(S,M)+E, where columns of S contain the source signals, M is the mixing matrix, and columns of E contain the noise signals. Columns of X are assumed to have zero mean. The goal is to find the unmixing matrix W such that columns of S=tcrossprod(X,W) are independent as possible.

Whitening Without loss of generality, we can write M=P%*%R where P is a tall matrix and R is an orthogonal rotation matrix. Letting Q denote the pseudoinverse of P, we can whiten the data using Y=tcrossprod(X,Q). The goal is to find the orthongal rotation matrix R such that the source signal estimates S=Y%*%R are as independent as possible. Note that W=crossprod(R,Q).

Infomax The Infomax approach finds the orthogonal rotation matrix R that (approximately) maximizes the joint entropy of a nonlinear function of the estimated source signals. See Bell and Sejnowski (1995) and Helwig (in prep) for specifics of algorithms.

Value

S	Matrix of source signal estimates (S=Y%*%R).
M	Estimated mixing matrix.
W	Estimated unmixing matrix (W=crossprod(R,Q)).
Y	Whitened data matrix.
Q	Whitening matrix.
R	Orthogonal rotation matrix.
vafs	Variance-accounted-for by each component.
iter	Number of algorithm iterations.
alg	Algorithm used (same as input).
fun	Contrast function (same as input).
signs	Component signs (same as input).
rate	Learning rate (same as input).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Bell, A.J. & Sejnowski, T.J. (1995). An information-maximization approach to blind separation and blind deconvolution. *Neural Computation, 7*, 1129-1159.

Examples

```
##########   EXAMPLE 1   ##########

# generate noiseless data (p==r)
set.seed(123)
nobs <- 1000
Amat <- cbind(icasamp("a","rnd",nobs),icasamp("b","rnd",nobs))
```

```
Bmat <- matrix(2*runif(4),2,2)
Xmat <- tcrossprod(Amat,Bmat)

# ICA via Infomax with 2 components
imod <- icaimax(Xmat,2)
acy(Bmat,imod$M)
cor(Amat,imod$S)



##########   EXAMPLE 2   ##########

# generate noiseless data (p!=r)
set.seed(123)
nobs <- 1000
Amat <- cbind(icasamp("a","rnd",nobs),icasamp("b","rnd",nobs))
Bmat <- matrix(2*runif(200),100,2)
Xmat <- tcrossprod(Amat,Bmat)

# ICA via Infomax with 2 components
imod <- icaimax(Xmat,2)
cor(Amat,imod$S)



##########   EXAMPLE 3   ##########

# generate noisy data (p!=r)
set.seed(123)
nobs <- 1000
Amat <- cbind(icasamp("a","rnd",nobs),icasamp("b","rnd",nobs))
Bmat <- matrix(2*runif(200),100,2)
Emat <- matrix(rnorm(10^5),1000,100)
Xmat <- tcrossprod(Amat,Bmat)+Emat

# ICA via Infomax with 2 components
imod <- icaimax(Xmat,2)
cor(Amat,imod$S)
```

`icajade` *ICA via JADE Algorithm*

Description

Computes ICA decomposition using Cardoso and Souloumiac's (1993, 1996) Joint Approximate Diagonalization of Eigenmatrices (JADE) approach.

Usage

```
icajade(X, nc, center = TRUE, maxit = 100,
        tol = 1e-6, Rmat = diag(nc))
```

Arguments

X	Data matrix with n rows (samples) and p columns (variables).
nc	Number of components to extract.
center	If TRUE, columns of X are mean-centered before ICA decomposition.
maxit	Maximum number of algorithm iterations to allow.
tol	Convergence tolerance.
Rmat	Initial estimate of the nc-by-nc orthogonal rotation matrix.

Details

ICA Model The ICA model can be written as X=tcrossprod(S,M)+E, where columns of S contain the source signals, M is the mixing matrix, and columns of E contain the noise signals. Columns of X are assumed to have zero mean. The goal is to find the unmixing matrix W such that columns of S=tcrossprod(X,W) are independent as possible.

Whitening Without loss of generality, we can write M=P%*%R where P is a tall matrix and R is an orthogonal rotation matrix. Letting Q denote the pseudoinverse of P, we can whiten the data using Y=tcrossprod(X,Q). The goal is to find the orthongal rotation matrix R such that the source signal estimates S=Y%*%R are as independent as possible. Note that W=crossprod(R,Q).

JADE The JADE approach finds the orthogonal rotation matrix R that (approximately) diagonalizes the cumulant array of the source signals. See Cardoso and Souloumiac (1993,1996) and Helwig and Hong (2013) for specifics of the JADE algorithm.

Value

S	Matrix of source signal estimates (S=Y%*%R).
M	Estimated mixing matrix.
W	Estimated unmixing matrix (W=crossprod(R,Q)).
Y	Whitened data matrix.
Q	Whitening matrix.
R	Orthogonal rotation matrix.
vafs	Variance-accounted-for by each component.
iter	Number of algorithm iterations.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Cardoso, J.F., & Souloumiac, A. (1993). Blind beamforming for non-Gaussian signals. *IEE Proceedings-F, 140*, 362-370.

Cardoso, J.F., & Souloumiac, A. (1996). Jacobi angles for simultaneous diagonalization. *SIAM Journal on Matrix Analysis and Applications, 17*, 161-164.

Helwig, N.E. & Hong, S. (2013). A critique of Tensor Probabilistic Independent Component Analysis: Implications and recommendations for multi-subject fMRI data analysis. *Journal of Neuroscience Methods, 213*, 263-273.

Examples

```
##########   EXAMPLE 1   ##########

# generate noiseless data (p==r)
set.seed(123)
nobs <- 1000
Amat <- cbind(icasamp("a","rnd",nobs),icasamp("b","rnd",nobs))
Bmat <- matrix(2*runif(4),2,2)
Xmat <- tcrossprod(Amat,Bmat)

# ICA via JADE with 2 components
imod <- icajade(Xmat,2)
acy(Bmat,imod$M)
cor(Amat,imod$S)



##########   EXAMPLE 2   ##########

# generate noiseless data (p!=r)
set.seed(123)
nobs <- 1000
Amat <- cbind(icasamp("a","rnd",nobs),icasamp("b","rnd",nobs))
Bmat <- matrix(2*runif(200),100,2)
Xmat <- tcrossprod(Amat,Bmat)

# ICA via JADE with 2 components
imod <- icajade(Xmat,2)
cor(Amat,imod$S)



##########   EXAMPLE 3   ##########

# generate noisy data (p!=r)
set.seed(123)
nobs <- 1000
Amat <- cbind(icasamp("a","rnd",nobs),icasamp("b","rnd",nobs))
Bmat <- matrix(2*runif(200),100,2)
Emat <- matrix(rnorm(10^5),1000,100)
Xmat <- tcrossprod(Amat,Bmat)+Emat
```

```
# ICA via JADE with 2 components
imod <- icajade(Xmat,2)
cor(Amat,imod$S)
```

`icaplot` *Plot Densities of Source Signal Distributions*

Description

Plot density (pdf) and kurtosis for the 18 source signal distributions used in Bach and Jordan (2002); see `icasamp` for more information.

Usage

```
icaplot(xseq = seq(-2,2,length.out=500),
        xlab = "", ylab = "", lty = 1,
        lwd = 1, col = "black", ...)
```

Arguments

`xseq`	Sequence of ordered data values for plotting density.
`xlab`	X-axis label for plot (default is no label).
`ylab`	Y-axis label for plot (default is no label).
`lty`	Line type for each density (scalar or vector of length 18).
`lwd`	Line width for each density (scalar or vector of length 18).
`col`	Line color for each density (scalar or vector of length 18).
`...`	Optional inputs for `plot`.

Value

Produces a plot with `NULL` return value.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Bach, F.R. (2002). *kernel-ica*. MATLAB toolbox (ver 1.2) http://www.di.ens.fr/~fbach/kernel-ica/.

Bach, F.R. & Jordan, M.I. (2002). Kernel independent component analysis. *Journal of Machine Learning Research, 3*, 1-48.

Examples

```
## Not run:
##########   EXAMPLE   ##########

quartz(height=9,width=7)
par(mar=c(3,3,3,3))
icaplot()

## End(Not run)
```

---

icasamp *Sample from Various Source Signal Distributions*

Description

Sample observations from the 18 source signal distributions used in Bach and Jordan (2002). Can also return density values and kurtosis for each distribution. Use `icaplot` to plot distributions.

Usage

```
icasamp(dname, query = c("rnd","pdf","kur"),
        nsamp = NULL, data = NULL)
```

Arguments

`dname`	Distribution name: letter "a" through "r" (see Bach & Jordan, 2002).
`query`	What to return: `query="rnd"` for random sample, `query="pdf"` for density values, and `query="kur"` for kurtosis.
`nsamp`	Number of observations to sample. Only used if `query="rnd"`.
`data`	Data values for density evaluation. Only used if `query="pdf"`.

Details

Inspired by `usr_distrib.m` from Bach's (2002) `kernel-ica` MATLAB toolbox.

Value

If `query="rnd"`, returns random sample of size `nsamp`.

If `query="pdf"`, returns density for input `data`.

If `query="kur"`, returns kurtosis of distribution.

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References

Bach, F.R. (2002). *kernel-ica*. MATLAB toolbox (ver 1.2) http://www.di.ens.fr/~fbach/kernel-ica/.

Bach, F.R. & Jordan, M.I. (2002). Kernel independent component analysis. *Journal of Machine Learning Research, 3*, 1-48.

Examples

```
##########   EXAMPLE   ##########

# sample 1000 observations from distribution "f"
set.seed(123)
mysamp <- icasamp("f","rnd",nsamp=1000)
xr <- range(mysamp)
hist(mysamp,freq=FALSE,ylim=c(0,.8),breaks=sqrt(1000))

# evaluate density of distribution "f"
xseq <- seq(-5,5,length.out=1000)
mypdf <- icasamp("f","pdf",data=xseq)
lines(xseq,mypdf)

# evaluate kurtosis of distribution "f"
icasamp("f","kur")
```

Index