Package ‘ica’

August 25, 2015

Type Package
Title Independent Component Analysis
Version 1.0-1
Date 2015-08-24
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 2015-08-25 00:51:02

R topics documented:

ica-package .................................................. 2
acy .......................................................... 3
congr ......................................................... 4
icafast ......................................................... 5
icaimax ....................................................... 8
icajade ....................................................... 10
icaplot ......................................................... 12
icasamp ..................................................... 14

Index 16
**Description**

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

**Details**

The functions `icafast`, `icaimax`, and `icajade` calculate ICA decompositions using the FastICA, Infomax, and JADE algorithms (respectively). The function `icasamp` can be used to sample from various interesting distributions, which are useful for comparing ICA algorithms.

**Author(s)**

Nathaniel E. Helwig <helwig@umn.edu>

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

**References**


**Examples**

# See examples for icafast, icaimax, icajade, and icasamp
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 $a_{ij} = (Y^{-1}X)_{ij}$.

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

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)

congru

Tucker’s Congruence Coefficient

Description

Calculates Tucker’s congruence coefficient (uncentered correlation) between x and y if these are vectors. If x and y are matrices then the congruence between the columns of x and y are computed.

Usage

congru(x, y = NULL)

Arguments

x Numeric vector, matrix or data frame.

y NULL (default) or a vector, matrix or data frame with compatible dimensions to x. The default is equivalent to y = x (but more efficient).

Details

Tucker’s congruence coefficient is defined as

\[ r = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \sum_{i=1}^{n} y_i^2}} \]

where \( x_i \) and \( y_i \) denote the \( i \)-th elements of \( x \) and \( y \).

Value

Returns a scalar or matrix with congruence coefficient(s).

Note

If x is a vector, you must also enter y.

Author(s)

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


Examples

```
set.seed(1)
A <- rnorm(100)
B <- rnorm(100)
C <- A*5
D <- A*(-0.5)
congru(A,B)
congru(A,C)
congru(A,D)
```

```
set.seed(1)
A <- cbind(rnorm(20),rnorm(20))
B <- cbind(A[,1]*-0.5,rnorm(20))
congru(A)
congru(A,B)
```

```R
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 \( x = \text{tcrossprod}(S, M) + E \), 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 \( S = \text{tcrossprod}(X, W) \) are independent as possible.

Whitening Without loss of generality, we can write \( M = P \times R \) 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 \( Y = \text{tcrossprod}(X, Q) \). The goal is to find the orthogonal rotation matrix \( R \) such that the source signal estimates \( S = Y \times R \) are as independent as possible. Note that \( W = \text{crossprod}(R, Q) \).

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

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

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 (\( S = Y \times R \)).
M
Estimated mixing matrix.
W
Estimated unmixing matrix (\( W = \text{crossprod}(R, Q) \)).
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


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$M)
congru(Amat,imod$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)
congru(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
```
icaimax <- icafast(Xmat,2)
congru(Amat,icaimax$S)

## icaimax  
**ICA via Infomax Algorithm**

### Description

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

### Usage

```r
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: \texttt{alg="newton"} for Newton iteration, and \texttt{alg="gradient"} for gradient descent.

- **fun**  
  Nonlinear (squashing) function to use for algorithm: \texttt{fun="tanh"} for hyperbolic tangent, \texttt{fun="log"} for logistic, and \texttt{fun="ext"} for extended Infomax.

- **signs**  
  Vector of length \( nc \) such that \( \text{signs}[j]=1 \) if \( j \)-th component is super-Gaussian and \( \text{signs}[j]=-1 \) if \( j \)-th component is sub-Gaussian. Only used if \texttt{fun="ext"}. Ignored if \texttt{signswitch=TRUE}.

- **signswitch**  
  If TRUE, the \texttt{signs} vector is automatically determined from the data; otherwise a confirmatory ICA decomposition is calculated using input \texttt{signs} vector. Only used if \texttt{fun="ext"}.

- **rate**  
  Learning rate for gradient descent algorithm. Ignored if \texttt{alg="newton"}.

- **rateanneal**  
  Annealing angle and proportion for gradient descent learning rate (see Details). Ignored if \texttt{alg="newton"}. 

Details

ICA Model The ICA model can be written as \( x = \text{tcrossprod}(S, M) + E \), 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 \( S = \text{tcrossprod}(X, W) \) 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 \( Y = \text{tcrossprod}(X, Q) \). The goal is to find the orthogonal rotation matrix \( R \) such that the source signal estimates \( S = Y * R \) are as independent as possible. Note that \( W = \text{crossprod}(R, Q) \).

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 \( (W = \text{crossprod}(R, Q)) \).
- \( Y \) Whitened data matrix.
- \( Q \) Whitening matrix.
- \( R \) Orthogonal rotation matrix.
- \( \text{vafs} \) Variance-accounted-for by each component.
- \( \text{iter} \) Number of algorithm iterations.
- \( \text{alg} \) Algorithm used (same as input).
- \( \text{fun} \) Contrast function (same as input).
- \( \text{signs} \) Component signs (same as input).
- \( \text{rate} \) Learning rate (same as input).

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References


Examples

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

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


### Usage

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

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

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 \( M = P \% \times R \) 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 orthogonal rotation matrix \( R \) such that the source signal estimates \( S = Y \% \times R \) are as independent as possible. Note that \( W = tcrossprod(R, Q) \).

**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

\begin{itemize}
  \item **S** Matrix of source signal estimates \( (S = Y \% \times R) \).
  \item **M** Estimated mixing matrix.
  \item **W** Estimated unmixing matrix \( (W = tcrossprod(R, Q)) \).
  \item **Y** Whitened data matrix.
  \item **Q** Whitening matrix.
  \item **R** Orthogonal rotation matrix.
  \item **vafs** Variance-accounted-for by each component.
  \item **iter** Number of algorithm iterations.
\end{itemize}

Author(s)

Nathaniel E. Helwig <helwig@umn.edu>

References


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$M)
congru(Amat,imod$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(100),100,2)
Xmat <- tcrossprod(Amat,Bmat)

# ICA via JADE with 2 components
imod <- icajade(Xmat,2)
congru(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(10^4),1000,100)
Xmat <- tcrossprod(Amat,Bmat)+Emat

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

icaplot  Plot Densities of Source Signal Distributions
**icaplot**

**Description**

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

**Usage**

```r
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**


**Examples**

```r
## 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

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

*Topic package
  ica-package, 2

acy, 3

congru, 4

ica (ica-package), 2
ica-package, 2
icafast, 2, 5
icaimax, 2, 8
icajade, 2, 10
icaplot, 12, 14
icasamp, 2, 13, 14