Package ‘steadyICA’

October 14, 2022

Type Package

Title ICA and Tests of Independence via Multivariate Distance Covariance

Version 1.0

Date 2015-11-08

Author Benjamin B. Risk and Nicholas A. James and David S. Matteson

Maintainer Benjamin Risk <bbr28@cornell.edu>

Description Functions related to multivariate measures of independence and ICA:
- estimate independent components by minimizing distance covariance;
- conduct a test of mutual independence based on distance covariance;
- estimate independent components via infomax (a popular method but generally performs poorer than mdcovica, ProDenICA, and/or fastICA, but is useful for comparisons);
- order independent components by skewness;
- match independent components from multiple estimates;
- other functions useful in ICA.

License GPL (>= 2)

Depends Rcpp (>= 0.9.13), MASS, clue, combinat

Suggests irlba, JADE, ProDenICA, fastICA, energy

LinkingTo Rcpp

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-11-11 00:08:41

R topics documented:

steadyICA-package .................................................. 2
compInd ................................................................. 3
dcovICA ............................................................... 4
dcovustat ............................................................. 5
frobICA ............................................................... 7
gmultidcov ........................................................... 9
steadyICA-package

Description

Functions related to multivariate measures of independence and ICA:
- estimate independent components by minimizing distance covariance;
- conduct a test of mutual independence based on distance covariance;
- estimate independent components via infomax (a popular method but generally performs poorer than steadyICA or ProDenICA but is useful for comparisons);
- order independent components by skewness;
- match independent components from multiple estimates;
- other functions useful in ICA.

Details

Package: steadyICA
Type: Package
Version: 1.0
Date: 2015-11-08
License: GPL (>= 2)
Depends: Rcpp (>= 0.9.13), MASS
Suggests: irlba, JADE, ProDenICA, fastICA

Author(s)

Benjamin B. Risk and Nicholas A. James and David S. Matteson.
Maintainer: Benjamin Risk <bbr28@cornell.edu>

References

**compInd**


See Also

`fastICA` `ProDenICA::ProDenICA`

Examples

```r
#see steadyICA
```

---

**Description**

Calculates a complete empirical measure of mutual multivariate independence. Makes use of the `utils::combn` function.

**Usage**

```r
compInd(S, group=1:ncol(S), alpha=1)
```

**Arguments**

- `S`: The `n x d` matrix for which you wish to calculate the dependence between `d` columns from `n` samples.
- `group`: A length `d` vector which indicates group membership for each component.
- `alpha`: The index used in calculating the distance between sample observations.

**Value**

Returns a scalar equal to the empirical multivariate distance between the observed samples, and their grouped counterpart.

**Note**

Suppose that the each component belongs to exactly one of `C` groups. This method makes use of the `utils::combn` and `combinat::permn` functions. As a result it will be both computationally and memory intensive, even for small to moderate `n` and small `C`.

**Author(s)**

Nicholas James
References


See Also

dcovustat, energy::dcov

Examples

library(steadyICA)
library(combinat)
set.seed(100)
S = matrix(rnorm(40),ncol=4)
group = c(1,2,3,3)
compInd(S,group,1)

dcovICA

ICA via distance covariance for 2 components

Description

This algorithm finds the rotation which minimizes the distance covariance between two orthogonal components via the angular parameterization of a 2x2 orthogonal matrix with the function stats::optimize. The results will be (approximately) equivalent to steadyICA but this function is much faster (but does not extend to higher dimensions).

Usage

dcovICA(Z, theta.0 = 0)

Arguments

Z The whitened n x d data matrix, where n is the number of observations and d the number of components.
theta.0 Determines the interval to be searched by the optimizer: lower bound = theta.0, upper bound = pi/2. Changing theta.0 affects the initial value, where the initial value = theta.0+(1/2+sqrt(5)/2)*pi/2, see optimize.

Value

theta.hat Estimated minimum.
W W = t(theta2W(theta.hat))
S Estimated independent components.
obj The distance covariance of S.
dcovustat

Calculate distance covariance via U-statistics

Author(s)

David Matteson and Benjamin Risk

References


See Also

steadyICA, optimize

Examples

library(JADE)
library(ProDenICA)
set.seed(123)
simS = cbind(rjordan(letter='j',n=1024),rjordan(letter='m',n=1024))
simM = mixmat(p=2)
xData = simS%*%simM
xWhitened = whitener(xData)

#Define true unmixing matrix as true M multiplied by the estimated whitener:
#Call this the target matrix:
W.true <- solve(simM%*%xWhitened$whitener)

a=Sys.time()
est.dCovICA = dcovICA(Z = xWhitened$Z,theta.0=0)
Sys.time()-a

#See the example with steadyICA for an explanation
#of the parameterization used in amari.error:
amari.error(t(est.dCovICA$W),W.true)

##NOTE: also try theta.0 = pi/4 since there may be local minima
## Not run: est.dcovICA = dcovICA(Z = xWhitened$Z,theta.0=pi/4)
amari.error(t(est.dcovICA$W),W.true)
## End(Not run)

a=Sys.time()
est.steadyICA = steadyICA(X=xWhitened$Z,verbose=TRUE)
Sys.time()-a
amari.error(t(est.steadyICA$W),W.true)
##theta parameterization with optimize is much faster
Description

Calculates the square of the U-statistic formulation of distance covariance. This is faster than the function 'dcov' in the R package 'energy' and requires less memory. Note that negative values are possible in this version.

Usage

dcovustat(x, y, alpha=1)

Arguments

x  A vector or matrix.
y  A vector or matrix with the same number of observations as x, though the number of columns of x and y may differ
alpha  A scaling parameter in the interval (0,2] used for calculating distances.

Value

Returns the distance covariance U-statistic.

Note

The value returned by dcovustat is equal to the square of the value returned by energy::dcov in the limit.

In dcovustat, a vector of length n is stored; in energy::dcov, an n x n matrix is stored. Thus, dcovustat requires far less memory and works for very large datasets.

Even though dcovustat converges to the square of the distance covariance of the random variables x and y, it can be negative.

Author(s)

David Matteson

References


See Also

multidcov, energy::dcov
Examples

```r
x = rnorm(5000)
y = rbinom(5000, 1, 0.5)
y = y - 1*(y==0)
z = y*exp(-x) # some non-linear dependence

dcovustat(x[1:1000], y[1:1000]) # close to zero

a = Sys.time()
dcovustat(x[1:1000], z[1:1000]) # greater than zero
a = Sys.time() - a

# measures of linear dependence close to zero:
cov(x, z)
cor(rank(x), rank(z))
```

```r
## Not run:
# dcovustat differs from energy::dcov but are equal in the limit
library(energy)
b = Sys.time()
(dcov(x[1:1000], z[1:1000]))^2
b = Sys.time() - b
as.double(b)/as.double(a) # dcovustat is much faster

## energy::dcov and dcovustat become approximately equal as n increases:
c = Sys.time()
dcovustat(x, z)
c = difftime(Sys.time(), c, sec)
d = Sys.time()
(dcov(x, z)^2)
d = difftime(Sys.time(), d, sec)
as.double(d)/as.double(c)

## End(Not run)
```

---

frobICA

**match mixing matrices or ICs and calculate their Frobenius distance**

Description

The ICA model is only identifiable up to signed permutations of the ICs. This function provides a similarity measure between two mixing matrices for the model \( X = S M + E \), where \( X \) is \( n \times p \), \( S \) is \( n \times d \), and \( M \) is \( d \times p \). The input is either two mixing matrices \( M_1 \) and \( M_2 \) or two matrices of independent components \( S_1 \) and \( S_2 \). For \( M_1 \) and \( M_2 \), `frobICA()` finds the signed row permutation of \( M_2 \) that minimizes the Frobenius norm between \( M_1 \) and \( M_2 \) using the Hungarian method. For \( S_1 \) and \( S_2 \), `frobICA()` finds the signed column permutation of \( S_2 \) that minimizes the Frobenius norm between \( S_1 \) and \( S_2 \). This function allows the mixing matrices (or independent components) to have differing numbers of rows (respectively, columns) such that the similarity measure is defined by the matching rows (resp., columns), and the non-matching rows (resp., columns) are discarded.
Usage

frobICA(M1 = NULL, M2 = NULL, S1 = NULL, S2 = NULL, standardize = FALSE)

Arguments

M1 A d x p mixing matrix
M2 A d x q mixing matrix
S1 An n x d matrix of independent components
S2 An n x q matrix of independent components
standardize Logical. See Note.

Details

frobICA(M1,M2) = 0 if there exists a signed permutation of the rows of M2 such that M1 = P%*%M2, where P is a d x q signed permutation matrix, i.e., composed of 0, 1, and -1, with d <= q; the function also allows d > q, in which case frobICA(M1,M2) = 0 if there exists a P such that P%*% M1 = M2. Unlike other ICA performance measures, this function can accommodate non-square mixing matrices.

Value

returns the Frobenius norm divided by p*min(d,q) (or n*min(d,q)) of the matched mixing matrices (resp., matched independent components).

Note

If standardize=TRUE, then scales the rows of M1 and M2 to have unit norm or the columns of S1 and S2 to have zero mean and sample variance equal to one. The user can supply either M1 and M2 or S1 and S2 but not both.

Author(s)

Benjamin Risk

References


See Also

JADE::MD clue::solve_LSAP matchICA
Examples

```r
mat1 <- matrix(rnorm(4*6),nrow=4)
perm <- matrix(c(-1,0,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1),4,4)
mat2 <- perm%*%mat1
sqrt(sum((mat1-mat2)^2))

frobICA(M1=mat1,M2=mat2)

# Another example showing invariance to permutations:
covMat <- t(mat1)%*%mat1
mvsample <- matrix(rnorm(400),100,4)%*%mat1
frobICA(M1=cov(mvsample),M2=covMat)
frobICA(M1=cov(mvsample),M2=covMat[sample(1:6),])

# Example using independent components:
nObs=300
simS<-cbind(rgamma(nObs, shape = 1, scale = 2),
             rgamma(nObs, shape = 3, scale = 2),
             rgamma(nObs, shape = 3, scale = 2),
             rgamma(nObs, shape = 9, scale = 0.5))

# Not necessary in this example, but this should be done when used with ICA:
simS <- apply(simS,2,scale)
frobICA(S1=simS,S2=simS%*%perm)

# Not run:
# returns an error if S1 and S2 are not explicitly defined:
frobICA(simS,simS%*%perm)

# End(Not run)
```

gmultidcov  

**Symmetric multivariate distance covariance for grouped components**

Description

Calculate either the symmetric or asymmetric multivariate distance covariance statistic for a given grouping of the components.

Usage

```r
gmultidcov(S,group=1:ncol(S),alpha=1,symmetric=TRUE)
```

Arguments

- `S` The n x d matrix for which you wish to calculate the dependence between d columns from n samples
- `group` A length d vector which indicates group membership for each component
alpha A scaling parameter in the interval (0,2] used for calculating distances.
symmetric logical; if TRUE (the default), calculates the symmetric version of the multivariate distance covariance. See details.

Details
Suppose that the groups are numbered 1,2,...,C and that group is a vector indicating group membership for each component. If symmetric==TRUE, calculates: sum_i=1^C d covustat(S[,group==i],S[,group!=i]) If symmetric==FALSE, calculates: sum_i=1^C-1 d covustat(S[,group==i],S[,group>i])

Value
Returns a scalar equal to the multivariate distance covariance statistic for grouped components of S.

Author(s)
Nicholas James

See Also
dcovustat, energy::dcov

Examples
library(steadyICA)
S = matrix(rnorm(300),ncol=3)
group = c(1,2,2)
gmultidcov(S,group,TRUE) # close to zero
gmultidcov(S,group,FALSE) # sill close to zero

Sigma = matrix(c(1,0.7,0,0.7,1,-0.2,0,-0.2,1),ncol=3)
X = MASS::mvrnorm(100,rep(0,3),Sigma)
gmultidcov(X,group,TRUE) # further from zero
gmultidcov(X,group,FALSE) # further from zero

infomaxICA

Estimates independent components via infomax

Description
Estimate independent components using the infomax criteria, which is equivalent to maximum likelihood using the logistic density, exp(-S)/(1+exp(-S))^2.

Usage
infomaxICA(X, n.comp, W.list = NULL, whiten = FALSE, maxit = 500, eps = 1e-08, alpha.eps = 1e-08, verbose = FALSE, restarts=0)
Arguments

- **X**: the n x p data matrix
- **n.comp**: number of components to be estimated
- **W.list**: list of orthogonal matrices for initialization
- **whiten**: Whitens the data before applying ICA, i.e., X%*%whitener = Z, where Z has mean zero and empirical covariance equal to the identity matrix, and Z is then used as the input.
- **maxit**: maximum number of iterations
- **eps**: algorithm terminates when the norm of the gradient is less than eps
- **alpha.eps**: tolerance controlling the level of annealing: algorithm terminates with a warning if the learning parameter is less than alpha.eps
- **verbose**: if TRUE, prints (1) the value of the infomax objective function at each iteration, (2) the norm of the gradient, and (3) current value of the learning parameter alpha.
- **restarts**: An integer determining the number of initial matrices to use in estimating the ICA model. The objective function has local optima, so multiple starting values are recommended. If whiten=TRUE, then generates random orthogonal matrices. If whiten=FALSE, generate random matrices from rnorm(). See code for details.

Details

This is an R version of ICA using the infomax criteria that provides an alternative to Matlab code ([ftp://ftp.cnl.salk.edu/pub/tony/sep96.public](ftp://ftp.cnl.salk.edu/pub/tony/sep96.public)), but with a few modifications. First, we use the full data (the so-called offline algorithm) in each iteration rather than an online algorithm with batches. Secondly, we use an adaptive method to choose the step size (based upon Bernaards and Jennrich 2005), which speeds up convergence. We also omitted the bias term (intercept) included in the original formulation because we centered our data.

Value

- **S**: the estimated independent components
- **W**: if whiten=TRUE, returns the orthogonal unmixing matrix; no value is returned when whiten=FALSE
- **M**: Returns the estimated mixing matrix for the model X = S M, where X is not pre-whitened (although X is centered)
- **f**: the value of the objective function at the estimated S
- **convergence**: summarizes algorithm status at each iteration
- **1** if norm of the gradient is less than eps, **2** if the learning parameter was smaller than alpha.eps, which usually means the gradient is sufficiently small, **0** otherwise

Note

In contrast to most other ICA methods, W is not constrained to be orthogonal.
Author(s)

Benjamin Risk

References


Examples

```r
## Example when p > d. The MD function and amari measures
# are not defined for M. We can compare the
# "true W inverse", which is the mixing matrix multiplied
# by the whitening matrix; alternatively, we can use
# multidcov::frobICA. These two approaches are
# demonstrated below:

set.seed(999)

nObs <- 1024
nComp <- 3

# simulate from gamma distributions with
# varying amounts of skewness:
simS<-cbind(rgamma(nObs, shape = 1, scale = 2),
           rgamma(nObs, shape = 3, scale = 2),
           rgamma(nObs, shape = 9, scale = 0.5))

#standardize by expected value and variance:
simS[,1] = (simS[,1] - 1*2)/sqrt(1*2^2)
simS[,2] = (simS[,2] - 3*2)/sqrt(3*2^2)
simS[,3] = (simS[,3] - 9*0.5)/sqrt(9*0.5^2)

# slightly revised 'mixmat' function (from ProDenICA)
# for p>=d: uses fastICA and ProDenICA parameterization:
myMixmat <- function (p = 2, d = NULL) {
  if(is.null(d)) d = p
  a <- matrix(rnorm(d * p), d, p)
  sa <- La.svd(a)
  dL <- sort(runif(d) + 1)
  mat <- sa$u%*%(sa$vt*dL)
  attr(mat, "condition") <- dL[d]/dL[1]
  mat
}

simM <- myMixmat(p = 6, d = nComp)
xData <- simS%*%simM
xWhitened <- whitener(xData, n.comp = nComp)

#Define a 'true' W (uses the estimated whitening matrix):
```
matchICA <- solve(simM %*% xWhitened$whitener)

estInfomax <- infomaxICA(X = xData, n.comp = nComp, whiten = TRUE, verbose = TRUE)

frobICA(estInfomax$M, simM)
library(JADE)
MD(t(estInfomax$W), t(solve(W.true)))
amari.error(t(estInfomax$W), t(solve(W.true)))

**matchICA**

match independent components using the Hungarian method

**Description**

The ICA model is only identifiable up to signed permutations of the ICs. This function finds the
signed permutation of a matrix S such that \|S^*P - template\| is minimized. Optionally also
matches the mixing matrix M.

**Usage**

matchICA(S, template, M = NULL)

**Arguments**

- **S**
  - the n x d matrix of ICs to be matched
- **template**
  - the n x d matrix that S is matched to.
- **M**
  - an optional d x p mixing matrix corresponding to S that will also be matched to the template

**Value**

Returns the signed permutation of S that is matched to the template. If the optional argument M is
provided, returns a list with the permuted S and M matrices.

**Author(s)**

Benjamin Risk

**References**

Kuhn, H. The Hungarian Method for the assignment problem Naval Research Logistics Quarterly,
1955, 2, 83 - 97

methods with an application to resting state fMRI.

**See Also**

frobICA clue::solve_LSAP
Examples

```r
set.seed(999)
nObs <- 1024
nComp <- 3
# simulate from some gamma distributions:
simS <- cbind(rgamma(nObs, shape = 1, scale = 2),
              rgamma(nObs, shape = 3, scale = 2),
              rgamma(nObs, shape = 9, scale = 0.5))
simM <- matrix(rnorm(9), 3)
pMat <- matrix(c(0, -1, 0, 1, 0, 0, 0, 0, -1), 3)
permS <- simS %*% pMat
permM <- t(pMat) %*% simM
matchedS <- matchICA(S = permS, template = simS, M = permM)
sum(abs(matchedS$S - simS))
sum(abs(simM - matchedS$M))
```

multidcov  

### Description

Calculate either the symmetric or asymmetric multivariate distance covariance statistic.

### Usage

```r
multidcov(S, symmetric = TRUE, alpha = 1)
```

### Arguments

- **S**: the n x d matrix for which you wish to calculate the dependence between d columns from n samples
- **alpha**: A scaling parameter in the interval (0, 2] used for calculating distances.
- **symmetric**: logical; if TRUE (the default), calculates the symmetric version of the multivariate distance covariance. See details.

### Details

If symmetric==TRUE, calculates: \( \sum_{i=1}^{d} \text{dcovustat}(S[,i], S[,-i]) \) If symmetric==FALSE, calculates: \( \sum_{i=1}^{d-1} \text{dcovustat}(S[,i], S[,i+1:d]) \)

### Value

returns a scalar equal to the multivariate distance covariance statistic for the columns of S

### Author(s)

David Matteson
permTest

See Also
dcovustat, energy::dcov

Examples

nObs <- 1024
nComp <- 3

simM <- matrix(rnorm(nComp*nComp),nComp)

# simulate some data:
simS<-cbind(rgamma(nObs, shape = 1, scale = 2),
            rgamma(nObs, shape = 3, scale = 2),
            rgamma(nObs, shape = 9, scale = 0.5))

simS <- scale(simS) #Standardize variance for identifiability

#mix the sources:
xData <- simS %*% simM

multidcov(simS) #close to zero
multidcov(whitener(xData)$Z) #should be larger than simS
multidcov(xData) #greater than zero

permTest

Permutation test for mutual independence.

Description

Calculates an approximate p-values based upon a permutation test for mutual independence.

Usage

permTest(S, group=1:ncol(S), R=199, FUN=c('gmultidcov','compInd'), ...)

Arguments

S The n x d matrix for which you wish to test the dependence between d columns from n samples

group A length d vector which indicates group membership for each component

R The number of permutations to perform in order to obtain the approximate p-value.

FUN The function used to determine mutual independence. This is one of either gmultidcov or compInd.

... Additional arguments passed to FUN. See details.
Details

Suppose that the groups are numbered 1, 2, ..., C and that group is a vector indicating group membership for each component. If symmetric==TRUE, calculates: \(\sum_{i=1}^{C} d\text{cov}_{\text{stat}}(S[,\text{group}\!=\!i],S[,\text{group}\!=\!i])\)

If symmetric==FALSE, calculates: \(\sum_{i=1}^{C-1} d\text{cov}_{\text{stat}}(S[,\text{group}\!=\!i],S[,\text{group}\!>\!i])\)

If no additional arguments are supplied for FUN then the default values are used. In the case of gmultidcov, values for alpha and symmetric can be supplied. While for compInd only the value of alpha is needed.

Value

Returns an approximate p-values based upon a permutation test.

Author(s)

Nicholas James

See Also

dcovustat, energy::dcov

dcov

dcovustat

Description

The ICA model is only identifiable up to signed permutations. This function provides a canonical ordering for ICA that is useful for fMRI or studies where signals are skewed. Multiplies columns of S that are left-skewed by -1 to force right skewness. Optionally orders the columns by descending skewness.

Usage

rightskew(S, M = NULL, order.skew = TRUE)

Arguments

S n x d matrix
M d x p mixing matrix
order.skew Option to return the permutation of columns of S from largest to smallest skewness. Also returns a permuted version of M that corresponds with the permuted S.

Value

Returns the matrix S such that all columns have positive skewness. If optional argument M is supplied, returns a list with the new S and corresponding M.
Author(s)
Benjamin Risk

References

Examples
nObs = 1024
simS<-cbind(rgamma(nObs, shape = 1, scale = 2),
            rgamma(nObs, shape = 9, scale = 0.5),
            -1*rgamma(nObs, shape = 3, scale = 2))
apply(simS,2,function(x){
  (sum((x - mean(x))^3)/length(x))/(sum((x - mean(x))^2)/length(x))^(3/2))
}
canonicalS <- rightskew(simS)
apply(canonicalS,2,function(x){
  (sum((x - mean(x))^3)/length(x))/(sum((x - mean(x))^2)/length(x))^(3/2))
}

steadyICA

Estimate independent components by minimizing distance covariance

Description
The model is: $X = S M + E$, where $X$ is $n \times p$ and has mean zero, $S$ is $n \times d$, $M$ is $d \times p$, and $E$ is measurement error. For whitened data, we have $Z = S t(W)$, where $W$ is orthogonal. We find the matrix $M$ such that $S$ minimizes the distance covariance dependency measure.

Usage
steadyICA(X, n.comp = ncol(X), w.init = NULL, PIT = FALSE, bw = 'SJ', adjust = 1,
whiten = FALSE, irlba = FALSE, symmetric = FALSE, eps = 1e-08, alpha.eps = 1e-08,
maxit = 100, method = c('Cpp','R'), verbose = FALSE)

Arguments
X The n x p data matrix, where n is the number of observations.
n.comp number of components to be estimated
w.init a p x d initial unmixing matrix
PIT logical: if TRUE, the distribution and density of the independent components are estimated using gaussian kernel density estimates.
bw Argument for bandwidth selection method; defaults to 'SJ'; see stats::density
steadyICA

**adjust**

Adjust bandwidth selection; e.g., if observations are correlated, consider using adjust > 1; see stats::density

**whiten**

Logical; if TRUE, whitens the data before applying ICA, i.e., \(X \times \text{whitener} = Z\), where \(Z\) has mean zero and empirical covariance equal to the identity matrix, and \(Z\) is then used as the input.

**irlba**

Logical; when whiten=TRUE, irlba=TRUE uses the R-package ‘irlba’ in the whitening, which is generally faster than base::svd though sometimes less accurate

**symmetric**

Logical; if TRUE, implements the symmetric version of the ICA algorithm, which is invariant to the ordering of the columns of \(X\) but is slower

**eps**

Algorithm terminates when the norm of the gradient of multidcov is less than eps

**maxit**

Maximum number of iterations

**alpha.eps**

Tolerance controlling the level of annealing: algorithm terminates with a warning if the learning parameter is less than alpha.eps

**method**

Options ‘Cpp’ (default), which requires the package ‘Rcpp’, or ‘R’, which is solely written in R but is much slower

**verbose**

Logical; if TRUE, prints the value of multidcov, norm of the gradient, and current value of the learning parameter.

**Value**

- \(S\) the estimated independent components
- \(W\) the estimated unmixing matrix: if whiten=TRUE, \(W\) is orthogonal and corresponds to \(ZW = S\); if whiten=FALSE, corresponds to \(X \text{ginv}(M) = S\)
- \(M\) Returns the estimated mixing matrix for the model \(X = S \ M\), where \(X\) is not pre-whitened (although \(X\) is centered)
- \(f\) the value of the objective function at the estimated \(S\)
- Table summarizes algorithm status at each iteration
- convergence 1 if norm of the gradient is less than eps, 2 if the learning parameter was smaller than alpha.eps, which usually means the gradient is sufficiently small, 0 otherwise

**Author(s)**

Benjamin Risk

**References**


**See Also**

multidcov
Examples

set.seed(999)
nObs <- 1024
nComp <- 3

# simulate from some gamma distributions:
simS <- cbind(rgamma(nObs, shape = 1, scale = 2),
              rgamma(nObs, shape = 3, scale = 2),
              rgamma(nObs, shape = 9, scale = 0.5))

# standardize by expected value and variance:
simS[,1] = (simS[,1] - 1*2)/sqrt(1*2^2)
simS[,2] = (simS[,2] - 3*2)/sqrt(3*2^2)
simS[,3] = (simS[,3] - 9*0.5)/sqrt(9*0.5^2)

# slightly revised 'mixmat' function (from ProDenICA)
# for p>=d: uses fastICA and ProDenICA parameterization:
myMixmat <- function (p = 2, d = NULL) {
  if(is.null(d)) d = p
  a <- matrix(rnorm(d * p), d, p)
  sa <- La.svd(a)
  dL <- sort(runif(d) + 1)
  mat <- sa$u%*%(sa$vt*dL)
  attr(mat, "condition") <- dL[d]/dL[1]
  mat
}

simM <- myMixmat(p = 6, d = nComp)
xData <- simS%*%simM
xWhitened <- whitener(xData, n.comp = nComp)

# estimate mixing matrix:
est.steadyICA.v1 = steadyICA(X = xData, whiten=TRUE, n.comp=nComp, verbose = TRUE)

# Define the 'true' W:
W.true <- solve(simM%*%xWhitened$whitener)
frobICA(M1=est.steadyICA.v1$M,M2=simM)
frobICA(S1=est.steadyICA.v1$S,S2=simS)

## Not run:
# now initiate from target:
est.steadyICA.v2 = steadyICA(X = xData, w.init= W.true, n.comp = nComp, whiten=TRUE, verbose=TRUE)

# estimate using PIT steadyICA such that dimension reduction is via ICA:
est.steadyICA.v3 = steadyICA(X = xData, w.init=ginv(est.steadyICA.v2$M),
PIT=TRUE, n.comp = nComp, whiten=FALSE, verbose=TRUE)

frobICA(M1=est.steadyICA.v2$M,M2=simM)
frobICA(M1=est.steadyICA.v3$M,M2=simM)
frobICA(S1=est.steadyICA.v2$S,S2=simS)

# tends to be lower than PCA-based (i.e., whitening) methods:
theta2W

Convert angles to an orthogonal matrix.

Description
Convert d*(d-1)/2 angles from a sequence of Givens rotations to a d x d orthogonal matrix.

Usage
theta2W(theta)

Arguments
theta A scalar or vector of length d*(d-1)/2 of values from which the d x d orthogonal matrix is calculated.

Value
A d x d orthogonal matrix resulting from the sequence of d*(d-1)/2 Givens rotation matrices.

Author(s)
David S. Matteson
W2theta

References


See Also

W2theta

Examples

# Generate orthogonal matrix:
mat <- matrix(rnorm(9), 3, 3)
W = svd(mat)$u

theta <- W2theta(W)

# Recovers W:
theta2W(theta)

W2theta

Convert an orthogonal matrix to its angular parameterization.

Description

Convert a d x d orthogonal matrix to a sequence of d*(d-1)/2 Givens rotations.

Usage

W2theta(W)

Arguments

W

A d x d orthogonal matrix.

Details

A d x d orthogonal matrix can be decomposed into a series of d*(d-1)/2 Givens rotation matrices, where each matrix is parameterized by a single angle.

Value

A vector of length d*(d-1)/2 comprised of the angles.

Author(s)

David S. Matteson

References

See Also

theta2W

Examples

theta = c(pi/6, pi/4, pi/2)

(W = theta2W(theta))

# Recover theta:
W2theta(W)

---

whitener

_Whitening function_

Description

Subtract column means and transform columns such that the empirical covariance is equal to the identity matrix. Uses the SVD.

Usage

whitener(X, n.comp = ncol(X), center.row = FALSE, irlba = FALSE)

Arguments

- **X** n x p matrix
- **n.comp** number of components to retain, i.e., first n.comp left eigenvectors from svd are retained
- **center.row** center both rows and columns prior to applying SVD (the resulting whitened data does not have zero-mean rows)
- **irlba** if TRUE, uses irlba to approximate the first n.comp left eigenvectors. See Note.

Value

- **whitener** the matrix such that X%*%whitener has zero mean and covariance equal to the identity matrix
- **Z** the whitened data, i.e., X%*%whitener = Z

Note

The use of the option 'irlba = TRUE' requires the package irlba and is very useful for large p. The function irlba only calculates the first n.comp eigenvectors and is much faster than svd for p >> n.comp, for e.g., in groupICA of fMRI data.
whitener

Author(s)

Benjamin Risk

See Also

svd, irlba::irlba

Examples

```r
simData <- cbind(rnorm(1000,1,2),rnorm(1000,-1,3),rnorm(1000,4,1))
simMVN <- simData%*%matrix(rnorm(12),3,4)
simWhiten <- whitener(simMVN,n.comp = 3)
colMeans(simWhiten$Z)
cov(simWhiten$Z)
```
Index

* **Givens**
  - $\theta_2W$, 20
  - $W_2\theta$, 21
* **ICA**
  - $\text{dcoVICA}$, 4
  - $\text{frobICA}$, 7
  - $\text{matchICA}$, 13
  - $\text{steadyICA}$, 17
* **Covariance**
  - $\text{compInd}$, 3
  - $\text{dcoVICA}$, 4
  - $\text{dcoVstat}$, 5
  - $\text{gmulticov}$, 9
  - $\text{multicov}$, 14
  - $\text{permTest}$, 15
  - $\text{steadyICA}$, 17
  - $\text{steadyICA-package}$, 2
* **dcoVICA**
  - $\text{steadyICA-package}$, 2
* **dcov**
  - $\text{compInd}$, 3
  - $\text{dcoVICA}$, 4
  - $\text{dcoVstat}$, 5
  - $\text{gmulticov}$, 9
  - $\text{multicov}$, 14
  - $\text{permTest}$, 15
* **Distance**
  - $\text{compInd}$, 3
  - $\text{dcoVICA}$, 4
  - $\text{dcoVstat}$, 5
  - $\text{frobICA}$, 7
  - $\text{gmulticov}$, 9
  - $\text{multicov}$, 14
  - $\text{permTest}$, 15
  - $\text{steadyICA}$, 17
  - $\text{steadyICA-package}$, 2
* **givens**
  - $\text{dcoVICA}$, 4
* **Hungarian**
  - $\text{matchICA}$, 13
* **ICA**
  - $\text{steadyICA-package}$, 2
* **Independent**
  - $\text{steadyICA}$, 17
* **Infomax**
  - $\text{steadyICA-package}$, 2
* **Matching**
  - $\text{steadyICA-package}$, 2
* **Orthogonal**
  - $\theta_2W$, 20
  - $W_2\theta$, 21
* **Prewhtien**
  - $\text{whitener}$, 22
* **Standardize**
  - $\text{whitener}$, 22
* **Whiten**
  - $\text{whitener}$, 22
* **clue::solve_LSAP**
  - 8, 13
  - $\text{compInd}$, 3
  - $\text{dcoVICA}$, 4
  - $\text{dcoVstat}$, 4, 5, 10, 15, 16
* **energy::dcov**
  - 4, 6, 10, 15, 16
* **fastICA**
  - 3
  - $\text{frobICA}$, 7, 13
* **gmulticov**
  - 9
* **InfomaxICA**
  - 10
* **irlba::irlba**
  - 23
* **JADE::MD**
  - 8
* **matchICA**
  - 8, 13
  - $\text{multicov}$, 6, 14, 18
* **optimize**
  - 4, 5
permTest, 15
ProDenICA::ProDenICA, 3

rightskew, 16

steadyICA, 5, 17
steadyICA-package, 2
svd, 23

theta2W, 20, 22

W2theta, 21, 21
whiten (whitener), 22
whitener, 22