Package 'ICtest'

February 16, 2021

Type Package

Title Estimating and Testing the Number of Interesting Components in Linear Dimension Reduction

Version 0.3-3

Date 2021-02-16

Maintainer Klaus Nordhausen <klaus.k.nordhausen@jyu.fi>

LinkingTo Rcpp, RcppArmadillo

Depends JADE, ICS (>= 1.3-0), ggplot2

Imports stats, graphics, Rcpp (>= 0.12.3), ICSNP, survey, GGally, png, zoo, xts

Description For different linear dimension reduction methods like principal components analysis (PCA), independent components analysis (ICA) and supervised linear dimension reduction tests and estimates for the number of interesting components (ICs) are provided.

License GPL (>= 2)

Suggests knitr, rmarkdown, fICA

VignetteBuilder knitr

NeedsCompilation yes

Author Klaus Nordhausen [aut, cre] (<https://orcid.org/0000-0002-3758-8501>), Hannu Oja [aut] (<https://orcid.org/0000-0002-4945-5976>), David E. Tyler [aut], Joni Virta [aut] (<https://orcid.org/0000-0002-2150-2769>)

Repository CRAN

Date/Publication 2021-02-16 17:10:23 UTC

R topics documented:

`components` *Generic Components Extraction Function*

Description

Function to extract components from an object. If the object is of class ictest or ladle the user can choose if all components are extracted or only those which were interesting under the null hypothesis.

Usage

```
components(x, ...)
## S3 method for class 'ictest'
components(x, which = "all", ...)
## S3 method for class 'ladle'
components(x, which = "all", ...)
```

Arguments

`x`	an object which has a components method, like for example an ictest object.
`which`	for an object of class ictest. If `"all"`, then all components S in the ictest object are extracted. If `"k"`, then only the first k components are extracted, where the value of `k` is taken from the ictest object. This is only meaningful if `k` was at least 1.
`...`	arguments passed on to other methods.

Value

a matrix with the components.

Author(s)

Klaus Nordhausen

Examples

```
n <- 200
X <- cbind(rnorm(n, sd = 2), rnorm(n, sd = 1.5), rnorm(n), rnorm(n), rnorm(n))

TestCov <- PCAasymp(X, k = 2)
head(components(TestCov))
head(components(TestCov, which = "k"))
```

`covSIR` *Supervised Scatter Matrix as Used in Sliced Inverse Regression*

Description

Sliced Inverse Regression (SIR) can be seen as special case of Supervised ICS (SICS) and this function gives the supervised scatter matrix for SIR

Usage

```
covSIR(X, y, h = 10, ...)
```

Arguments

`X`	a numeric data matrix.
`y`	a numeric response vector.
`h`	the number of slices.
`...`	arguments passed on to `quantile`.

Details

This supervised scatter matrix is usually used as the second scatter matrix in SICS to obtain a SIR type supervised linear dimension reduction. For that purpose `covSIR` first divides the response `y` into `h` slices using the corresponding quantiles as cut points. Then for each slice the mean vector of `X` is computed and the resulting supervised scatter matrix consist of the covariance matrix of these mean vectors.

The function might have problems if the sample size is too small.

Value

a supervised scatter matrix

Author(s)

Klaus Nordhausen

References

Liski, E., Nordhausen, K. and Oja, H. (2014), Supervised invariant coordinate selection, Statistics: A Journal of Theoretical and Applied Statistics, ***48***, *711–731. <doi:10.1080/02331888.2013.800067>.*

Nordhausen, K., Oja, H. and Tyler, D.E. (2016), Asymptotic and bootstrap tests for subspace dimension, <https://arxiv.org/abs/1611.04908>.

See Also

`ics`

Examples

```
X <- matrix(rnorm(1000), ncol = 5)
eps <- rnorm(200, sd = 0.1)
y <- 2 + 0.5 * X[, 1] + 2 * X[, 3] + eps

covSIR(X, y)
```

`FOBIasymp` *Testing for the Number of Gaussian Components in NGCA or ICA Using FOBI*

Description

In non-gaussian component analysis (NGCA) and independent components analysis (ICA) gaussian components are considered as uninteresting. The function tests, based on FOBI, if there are `p-k` gaussian components where `p` is the dimension of the data. The function offers three different test versions.

Usage

```
FOBIasymp(X, k, type = "S3", model = "NGCA", method = "satterthwaite")
```

Arguments

X	numeric data matrix.
k	the number of non-gaussian components under the null.
type	which of the three tests to perform. Options are "S1", "S2" and "S3". For the differences see the details section.
model	What is the underlying assumption of the non-gaussian parts. Options are general "NGCA" model and "ICA" model.
method	if type = "S1" the teststatistic has as limiting distribution a weighted sum of chisquare distributions. To compute the p-value then the function used is pchisqsum. The method argument specifies which method pchisqsum uses for the computation. Options are "satterthwaite", "integration" and "saddlepoint".

Details

The function jointly diagonalizes the regular covariance and the matrix of fourth moments. Note however that in this case the matrix of fourth moments is not made consistent under the normal model by dividing it by $p+2$, as for example done by the function cov4 where p denotes the dimension of the data. Therefore the eigenvalues of this generalized eigenvector-eigenvalue problem which correspond to normally distributed components should be p+2.

Given eigenvalues $d_1, ..., d_p$ the function thus orders the components in decending order according to the values of $(d_i - (p+2))^2$.

Under the null it is then assumed that the first k interesting components are mutually independent and non-normal and the last p-k are gaussian.

Three possible tests are then available to test this null hypothesis for a sample of size n:

1. type="S1": The test statistic T is the variance of the last p-k eigenvalues around p+2:

$$T = n \sum_{i=k+1}^{p} (d_i - (p+2))^2$$

the limiting distribution of which under the null is the sum of two weighted chisquare distributions with weights:
$w_1 = 2\sigma_1/(p-k)$ and $w_2 = 2\sigma_1/(p-k) + \sigma_2$.
and degrees of freedom:
$df_1 = (p-k-1)(p-k+2)/2$ and $df_2 = 1$.

2. type="S2": Another possible version for the test statistic is a scaled sum of the variance of the eigenvalues around the mean plus the variance around the expected value under normality (p+2). Denote VAR_{dpk} as the variance of the last p-k eigenvalues and $VAR2_{dpk}$ as the variance of these eigenvalues around $p+2$. Then the test statistic is:

$$T = (n(p-k)VAR_{dpk})/(2\sigma_1) + (nVAR2_{dpk})/(2\sigma_1/(p-k) + \sigma_2)$$

This test statistic has a limiting chisquare distribution with $(p-k-1)(p-q+2)/2 + 1$ degrees of freedom.

3. `type="S3"`: The third possible test statistic just checks the equality of the last p-k eigenvalues using only the first part of the test statistic of `type="S2"`. The test statistic is then:

$$T = (n(p-k)VAR_{dpk})/(2\sigma_1)$$

and has a limiting chisquare distribution with $(p-k-1)(p-q+2)/2$ degrees of freedom.

The constants σ_1 and σ_2 depend on the underlying model assumptions as specified in argument `model` and are estimated from the data.

Value

A list of class ictest inheriting from class htest containing:

`statistic`	the value of the test statistic.
`p.value`	the p-value of the test.
`parameter`	the degrees of freedom of the test or the degrees of freedoms and the corresponding weights of the test in case the test has as its limiting distribution a weighted sum of chisquare distributions.
`method`	character string denoting which test was performed.
`data.name`	character string giving the name of the data.
`alternative`	character string specifying the alternative hypothesis.
`k`	the number or non-gaussian components used in the testing problem.
`W`	the transformation matrix to the independent components. Also known as unmixing matrix.
`S`	data matrix with the centered independent components.
`D`	the underlying FOBI eigenvalues.
`MU`	the location of the data which was substracted before calculating the independent components.
`sigma1`	the asymptotic constant sigma1 needed for the asymptotic test(s).
`sigma2`	the asymptotic constant sigma2 needed for the asymptotic test(s).
`type`	the value of `type`.
`model`	the value of `model`.

Author(s)

Klaus Nordhausen

References

Nordhausen, K., Oja, H. and Tyler, D.E. (2016), Asymptotic and bootstrap tests for subspace dimension, <https://arxiv.org/abs/1611.04908>.

Nordhausen, K., Oja, H., Tyler, D.E. and Virta, J. (2017), Asymptotic and Bootstrap Tests for the Dimension of the Non-Gaussian Subspace, Signal Processing Letters, 24, 887–891. <doi:10.1109/LSP.2017.2696880 >.

See Also

FOBI, FOBIboot

Examples

```
n <- 1500
S <- cbind(runif(n), rchisq(n, 2), rexp(n), rnorm(n), rnorm(n), rnorm(n))
A <- matrix(rnorm(36), ncol = 6)
X <- S %*% t(A)

FOBIasymp(X, k = 2)
FOBIasymp(X, k = 3, type = "S1")
FOBIasymp(X, k = 0, type = "S2", model = "ICA")
```

FOBIboot *Boostrap-based Testing for the Number of Gaussian Components in ICA Using FOBI*

Description

In independent components analysis (ICA) gaussian components are considered as uninteresting. The function uses boostrappping tests, based on FOBI, to decide if there are p-k gaussian components where p is the dimension of the data. The function offers two different boostrapping strategies.

Usage

```
FOBIboot(X, k, n.boot = 200, s.boot = "B1")
```

Arguments

X	a numeric data matrix with p>1 columns.
k	the number of non-gaussian components under the null.
n.boot	number of bootstrapping samples.
s.boot	bootstrapping strategy to be used. Possible values are "B1", "B2". See details for further information.

Details

As in FOBIasymp the function jointly diagonalizes the regular covariance and the matrix of fourth moments. Note that in this case the matrix of fourth moments is not made consistent under the normal model by dividing it by $p + 2$, as for example done by the function cov4 where p denotes the dimension of the data. Therefore the eigenvalues of this generalized eigenvector-eigenvalue problem which correspond to normally distributed components should be p+2. Given eigenvalues $d_1, ..., d_p$ the function thus orders the components in descending order according to the values of $(d_i - (p + 2))^2$.

Under the null it is then assumed that the first k interesting components are mutually independent and non-normal and the last p-k components are gaussian.

Let $d_1, ..., d_p$ be the ordered eigenvalues, W the correspondingly ordered unmixing matrix, $s_i = W(x_i - MU)$ the corresponding source vectors which give the source matrix S which can be decomposed into S_1 and S_2 where S_1 is the matrix with the k non-gaussian components and S_2 the matrix with the gaussian components (under the null).

The test statistic is then $T = n \sum_{i=k+1}^{p} (d_i - (p+2))^2$

Two possible bootstrap tests are provided for testing that the last `p-k` components are gaussian and independent from the first k components:

1. `s.boot="B1"`: The first strategy has the followong steps:
   (a) Take a bootstrap sample S_1^* of size n from S_1.
   (b) Take a bootstrap sample S_2^* consisting of a matrix of standard normally distributed elements.
   (c) Combine $S^* = (S_1^*, S_2^*)$ and create $X^* = S^*W$.
   (d) Compute the test statistic based on X^*.
   (e) Repeat the previous steps `n.boot` times.

   Note that in this bootstrapping test the assumption of "independent components" is not used, it is only used that the last $p - k$ components are gaussian and independent from the first k components. Therefore this strategy can be applied in an independent component analysis (ICA) framework and in a non-gaussian components analysis (NGCA) framework.
2. `s.boot="B2"`: The second strategy has the following steps:
   (a) Take a bootstrap sample S_1^* of size n from S_1 where the subsampling is done separately for each independent component.
   (b) Take a bootstrap sample S_2^* consisting of a matrix of standard normally distributed elemenets.
   (c) Combine $S^* = (S_1^*, S_2^*)$ and create $X^* = S^*W$.
   (d) Compute the test statistic based on X^*.
   (e) Repeat the previous steps `n.boot` times.

   This bootstrapping strategy assumes a full ICA model and cannot be used in an NGCA framework.

Value

A list of class ictest inheriting from class htest containing:

`statistic`	the value of the test statistic.
`p.value`	the p-value of the test.
`parameter`	the number of boostrapping samples used to obtain the p-value.
`method`	character string which test was performed.
`data.name`	character string giving the name of the data.
`alternative`	character string specifying the alternative hypothesis.
`k`	the number or non-gaussian components used in the testing problem.
`W`	the transformation matrix to the independent components. Also known as unmixing matrix.

S	data matrix with the centered independent components.
D	the underlying FOBI eigenvalues.
MU	the location of the data which was substracted before calculating the independent components.
s.boot	character string which boostrapping strategy was used.

Author(s)

Klaus Nordhausen

References

Nordhausen, K., Oja, H. and Tyler, D.E. (2016), Asymptotic and bootstrap tests for subspace dimension, <https://arxiv.org/abs/1611.04908>.

Nordhausen, K., Oja, H., Tyler, D.E. and Virta, J. (2017), Asymptotic and Bootstrap Tests for the Dimension of the Non-Gaussian Subspace, Signal Processing Letters, 24, 887–891. <doi:10.1109/LSP.2017.2696880 >.

See Also

FOBI, FOBIasymp

Examples

```
n <- 1500
S <- cbind(runif(n), rchisq(n, 2), rexp(n), rnorm(n), rnorm(n), rnorm(n))
A <- matrix(rnorm(36), ncol = 6)
X <- S %*% t(A)

FOBIboot(X, k = 2)
FOBIboot(X, k = 3, s.boot = "B1")
FOBIboot(X, k = 0, s.boot = "B2")
```

FOBIladle *Ladle Estimate to Estimate the Number of Gaussian Components in ICA or NGCA*

Description

The ladle estimator uses the eigenvalues and eigenvectors of FOBI to estimate the number of Gaussian components in ICA or NGCA.

Usage

```
FOBIladle(X, n.boot = 200,
          ncomp = ifelse(ncol(X) > 10, floor(ncol(X)/log(ncol(X))), ncol(X) - 1))
```

Arguments

X	numeric data matrix.
n.boot	number of bootstrapping samples to be used.
ncomp	The number of components among which the ladle estimator is to be searched. The default here follows the recommendation of Luo and Li 2016.

Details

The model here assumes that in ICA or NGCA there are k non-gaussian components and p-k gaussian components. The idea is then to decide which eigenvalues differ from p+2. The ladle estimate for this purpose combines the values of the scaled eigenvalues and the variation of the eigenvectors based on bootstrapping. The idea there is that for distinct eigenvales the variation of the eigenvectors is small and for equal eigenvalues the corresponding eigenvectors have large variation.

This measure is then computed assuming k=0,..., ncomp and the ladle estimate for k is the value where the measure takes its minimum.

Value

A list of class ladle containing:

method	the string FOBI.
k	the estimated number of non-gaussian components.
fn	vector giving the measures of variation of the eigenvectors using the bootstrapped eigenvectors for the different number of components.
phin	normalized eigenvalues of the FOBI matrix.
gn	the main criterion for the ladle estimate - the sum of fn and phin. k is the value where gn takes its minimum
lambda	the eigenvalues of the FOBI matrix.
W	the transformation matrix to the independent components. Also known as unmixing matrix.
S	data matrix with the centered independent components.
MU	the location of the data which was substracted before calculating the independent components.
data.name	the name of the data for which the ladle estimate was computed.

Author(s)

Klaus Nordhausen

References

Luo, W. and Li, B. (2016), Combining Eigenvalues and Variation of Eigenvectors for Order Determination, Biometrika, 103. 875–887. <doi:10.1093/biomet/asw051>

See Also

ladleplot

Examples

```
n <- 1000
X <- cbind(rexp(n), rt(n,5), rnorm(n), rnorm(n), rnorm(n), rnorm(n))

test <- FOBIladle(X)
test
summary(test)
plot(test)
ladleplot(test)
```

---

ggladleplot *Ladle Plot for an Object of Class ladle Using ggplot2*

Description

The ladle plot is a measure to decide about the number of interesting components. Of interest for the ladle criterion is the minimum. The function here offers however also to plot other criterion values which are part of the actual ladle criterion.

Usage

```
ggladleplot(x, crit = "gn", type="l", ylab = crit,
          xlab = "component", main = deparse(substitute(x)), ...)
```

Arguments

x	an object of class ladle.
crit	the criterion to be plotted, options are "gn", "fn", "phin" and "lambda".
type	plotting type.
ylab	default ylab value.
xlab	default xlab value.
main	default title.
...	other arguments for the plotting functions.

Details

The main criterion of the ladle is the scaled sum of the eigenvalues and the measure of variation of the eigenvectors up to the component of interest.

The sum is denoted "gn" and the individual parts are "fn" for the measure of the eigenvector variation and "phin" for the scaled eigenvalues. The last option "lambda" corresponds to the unscaled eigenvalues yielding then a screeplot.

Author(s)

Klaus Nordhausen, Joni Virta

References

Luo, W. and Li, B. (2016), Combining Eigenvalues and Variation of Eigenvectors for Order Determination, Biometrika, 103. 875–887. <doi:10.1093/biomet/asw051>

See Also

FOBIladle, PCAladle, SIRladle

Examples

```
n <- 1000
X <- cbind(rexp(n), rt(n,5), rnorm(n), rnorm(n), rnorm(n), rnorm(n))
test <- FOBIladle(X)
ggladleplot(test)
ggladleplot(test, crit="fn")
ggladleplot(test, crit="phin")
ggladleplot(test, crit="lambda")
```

---

ggplot.ictest *Scatterplot Matrix for a ictest Object using ggplot2*

Description

For an object of class ictest, plots either the pairwise scatter plot matrix using `ggpairs` from GGally, or the time series plots of the underlying components using ggplot2. The user can choose if only the components considered interesting or all of them should be plotted. Aesthetics can be passed to ggpairs as well.

Usage

```
## S3 method for class 'ictest'
ggplot(data, mapping = aes(), mapvar = NULL, which = "all", ...,
       environment=parent.frame())
```

Arguments

`data`	object of class ictest
`mapping`	aesthetic mapping, see documentation for `ggpairs`. If `x` has the class `mts` then this argument is not used.
`mapvar`	data.frame of the external variables used by the aesthetic mappings. If `x` has the class `mts` then this argument is not used.

which	if "all", then all components of S in the ictest object are plotted. If "k", then only the first k components are plotted, where the value of k is taken from the ictest object. This is only meaningful if k was at least 2.
...	arguments passed on to ggpairs. If the component matrix has the class mts, xts or zoo then this argument is not used.
environment	not used but needed for consistency.

Details

If the component matrix has the class mts, xts or zoo then a time series plot will be plotted using ggplot2. Otherwise, a pairwise scatter plot matrix will be plotted using GGally.

Author(s)

Klaus Nordhausen, Joni Virta

See Also

plot.ictest,pairs

Examples

```
# The data
X <- iris[, 1:4]

# The aesthetics variables
mapvar <- data.frame(iris[, 5])
colnames(mapvar) <- "species"

TestCov <- PCAasymp(X, k = 2)
ggplot(TestCov)
ggplot(TestCov, aes(color = species), mapvar = mapvar, which = "k")
```

---

ggplot.ladle *Scatterplot Matrix for a ladle Object using ggplot2*

Description

For an object of class ladle, plots either the pairwise scatter plot matrix using ggpairs from GGally, or the time series plots of the underlying components using ggplot2. The user can choose if only the components considered interesting or all of them should be plotted. Aesthetics can be passed to ggpairs as well.

Usage

```
## S3 method for class 'ladle'
ggplot(data, mapping = aes(), mapvar = NULL, which = "all", ...,
       environment=parent.frame())
```

Arguments

data	object of class ladle
mapping	aesthetic mapping, see documentation for ggpairs. If x has the class mts then this argument is not used.
mapvar	data.frame of the external variables used by the aesthetic mappings. If x has the class mts then this argument is not used.
which	if "all", then all components of S in the ladle object are plotted. If "k", then only the first k components are plotted, where the value of k is taken from the ladle object. This is only meaningful if k was at least 2.
...	arguments passed on to ggpairs. If the component matrix has the class mts, xts or zoo then this argument is not used.
environment	not used but needed for consistency.

Details

If the component matrix has the class mts, xts or zoo then a time series plot will be plotted using ggplot2. Otherwise, a pairwise scatter plot matrix will be plotted using GGally.

Author(s)

Klaus Nordhausen, Joni Virta

See Also

plot.ladle,pairs

Examples

```
# The data
X <- as.matrix(iris[, 1:4])

# The aesthetics variables
mapvar <- data.frame(iris[, 5])
colnames(mapvar) <- "species"

ladle_res <- PCAladle(X)

# The estimate
summary(ladle_res)

# Plots of the components
ggplot(ladle_res)
ggplot(ladle_res, aes(color = species), mapvar = mapvar, which = "k")
```

ggscreeplot *Screeplot for an ictest Object Using ggplot2*

Description

Plots the criterion values of an `ictest` object against its index number using ggplot2. Two versions of this screeplot are available.

Usage

```
ggscreeplot(x, type = "barplot", main = deparse(substitute(x)),
            ylab = "criterion", xlab = "component")
```

Arguments

`x`	object of class `ictest`.
`type`	`barplot` if a barplot or `lines` if a line plot is preferred.
`main`	main title of the plot.
`ylab`	y-axis label.
`xlab`	x-axis label.

Author(s)

Klaus Nordhausen, Joni Virta

See Also

`screeplot.ictest`

Examples

```
n <- 200
X <- cbind(rnorm(n, sd = 2), rnorm(n, sd = 1.5), rnorm(n), rnorm(n), rnorm(n))

TestCov <- PCAasymp(X, k = 2)
ggscreeplot(TestCov)
```

`ladle` *Ladle estimate for an arbitrary matrix*

Description

The ladle estimates the rank of a symmetric matrix S by combining the classical screeplot with an estimate of the rank from the bootstrap eigenvector variability of S.

Usage

```
ladle(x, S, n.boots = 200, ...)
```

Arguments

`x`	n x p data matrix.
`S`	Function for computing a q x q symmetric matrix from the data `x`.
`n.boots`	The number of bootstrap samples.
`...`	Furhter parameters passed to `S`

Details

Assume that the eigenvalues of the population version of `S` are $\lambda_1 >= ... >= \lambda_k > \lambda_k + 1 = ... = \lambda_p$. The ladle estimates the true value of k (for example the rank of `S`) by combining the classical screeplot with estimate of k from the bootstrap eigenvector variability of `S`.

For applying the ladle to either PCA, FOBI or SIR, see the dedicated functions `PCAladle`, `FOBIladle`, `SIRladle`.

Value

A list of class `ladle` containing:

`method`	The string "general".
`k`	The estimated value of k.
`fn`	A vector giving the measures of variation of the eigenvectors using the bootstrapped eigenvectors for the different number of components.
`phin`	The normalized eigenvalues of the S matrix.
`gn`	The main criterion for the ladle estimate - the sum of fn and phin. k is the value where gn takes its minimum.
`lambda`	The eigenvalues of the covariance matrix.
`data.name`	The name of the data for which the ladle estimate was computed.

Author(s)

Joni Virta

References

Luo, W. and Li, B. (2016), Combining Eigenvalues and Variation of Eigenvectors for Order Determination, Biometrika, 103. 875-887. <doi:10.1093/biomet/asw051>

See Also

PCAladle, FOBIladle, SIRladle

Examples

```
# Function for computing the left CCA matrix
S_CCA <- function(x, dim){
  x1 <- x[, 1:dim]
  x2 <- x[, -(1:dim)]
  stand <- function(x){
    x <- as.matrix(x)
    x <- sweep(x, 2, colMeans(x), "-")
    eigcov <- eigen(cov(x), symmetric = TRUE)
    x%*%(eigcov$vectors%*%diag((eigcov$values)^(-1/2))%*%t(eigcov$vectors))
  }

  x1stand <- stand(x1)
  x2stand <- stand(x2)

  crosscov <- cov(x1stand, x2stand)

  tcrossprod(crosscov)
}

# Toy data with two canonical components
n <- 200
x1 <- matrix(rnorm(n*5), n, 5)
x2 <- cbind(x1[, 1] + rnorm(n, sd = sqrt(0.5)),
            -1*x1[, 1] + x1[, 2] + rnorm(n, sd = sqrt(0.5)),
            matrix(rnorm(n*3), n, 3))
x <- cbind(x1, x2)

# The ladle estimate
ladle_1 <- ladle(x, S_CCA, dim = 5)
ladleplot(ladle_1)
```

`ladleplot` *Ladle Plot for an Object of Class ladle*

Description

The ladle plot is a measure to decide about the number of interesting components. Of interest for the ladle criterion is the minimum. The function here offers however also to plot other criterion values which are part of the actual ladle criterion.

Usage

```
ladleplot(x, crit = "gn", type="l", ylab = crit,
          xlab = "component", main = deparse(substitute(x)), ...)
```

Arguments

x	an object of class ladle.
crit	the criterion to be plotted, options are "gn", "fn", "phin" and "lambda".
type	plotting type.
ylab	default ylab value.
xlab	default xlab value.
main	default title.
...	other arguments for the plotting functions.

Details

The main criterion of the ladle is the scaled sum of the eigenvalues and the measure of variation of the eigenvectors up to the component of interest.

The sum is denoted "gn" and the individual parts are "fn" for the measure of the eigenvector variation and "phin" for the scaled eigenvalues. The last option "lambda" corresponds to the unscaled eigenvalues yielding then a screeplot.

Author(s)

Klaus Nordhausen

References

Luo, W. and Li, B. (2016), Combining Eigenvalues and Variation of Eigenvectors for Order Determination, Biometrika, 103. 875–887. <doi:10.1093/biomet/asw051>

See Also

FOBIladle, PCAladle, SIRladle

Examples

```
n <- 1000
X <- cbind(rexp(n), rt(n,5), rnorm(n), rnorm(n), rnorm(n), rnorm(n))
test <- FOBIladle(X)
ladleplot(test)
ladleplot(test, crit="fn")
ladleplot(test, crit="phin")
ladleplot(test, crit="lambda")
```

NGPP *Non-Gaussian Projection Pursuit*

Description

Estimates k non-Gaussian signal components using projection pursuit. The projection index can be chosen among convex combinations of squares of one or several standard projection indices used in ICA.

Usage

```
NGPP(X, k, nl = c("skew", "pow3"), alpha = 0.8, method = "symm", eps = 1e-6,
     verbose = FALSE, maxiter = 100)
```

Arguments

X	Numeric matrix with n rows corresponding to the observations and p columns corresponding to the variables.
k	Number of components to estimate, $1 \leq k \leq p$.
nl	Vector of non-linearities, a convex combination of the corresponding squared objective functions of which is then used as the projection index. The choices include "skew" (skewness), "pow3" (excess kurtosis), "tanh" ($log(cosh)$) and "gauss" (Gaussian function).
alpha	Vector of positive weights between 0 and 1 given to the non-linearities. The length of alpha should be either one less than the number of non-linearities in which case the missing weight is chosen so that alpha sums to one, or equal to the number of non-linearities in which case the weights are used as such. No boundary checks for the weights are done.
method	If "symm" the k signals are estimated simultaneously (symmetric projection pursuit) and if "defl" they are estimated one-by-one (deflation-based projection pursuit).
eps	Convergence tolerance.
verbose	If TRUE the numbers of iterations will be printed.
maxiter	Maximum number of iterations.

Details

It is assumed that the data is a random sample from the model $x = m + As$ where the latent vector $s = (s_1^T, s_2^T)^T$ consists of k-dimensional non-Gaussian subvector (the signal) and $p-k$-dimensional Gaussian subvector (the noise) and the components of s are mutually independent. Without loss of generality we further assume that the components of s have zero means and unit variances.

The objective is to estimate an inverse for the mixing matrix A and in non-Gaussian projection pursuit this is done by first standardizaing the observations and then finding mutually orthogonal directions maximizing a convex combination of the chosen squared objective functions.

After estimation the found signals are ordered in decreasing order with respect to their objective function values.

Value

A list with class 'bss' containing the following components:

W	Estimated unmixing matrix
S	Matrix of size $n \times k$ containing the estimated signals.
D	Vector of the objective function values of the signals
MU	Location vector of the data which was substracted before estimating the signal components.

Author(s)

Joni Virta

References

Virta, J., Nordhausen, K. and Oja, H., (2016), Projection Pursuit for non-Gaussian Independent Components, <https://arxiv.org/abs/1612.05445>.

See Also

NGPPsim,NGPPest,fICA

Examples

```
# Simulated data with 2 signals

n <- 500
S <- cbind(rexp(n), runif(n),  rnorm(n))
A <- matrix(rnorm(9), ncol = 3)
X <- S %*% t(A)

res <- NGPP(X, 2)
res$W %*% A


# Iris data

X <- as.matrix(iris[, 1:4])

res <- NGPP(X, 2, nl = c("pow3", "tanh"), alpha = 0.5)
plot(res, col = iris[, 5])
```

NGPPest *Signal Subspace Dimension Testing Using non-Gaussian Projection Pursuit*

Description

Estimates the dimension of the signal subspace using NGPP to conduct sequential hypothesis testing. The test statistic is a multivariate extension of the classical Jarque-Bera statistic and the distribution of it under the null hypothesis is obtained by simulation.

Usage

```
NGPPest(X, nl = c("skew", "pow3"), alpha = 0.8, N = 500, eps = 1e-6,
        verbose = FALSE, maxiter = 100)
```

Arguments

X	Numeric matrix with n rows corresponding to the observations and p columns corresponding to the variables.
nl	Vector of non-linearities, a convex combination of the corresponding squared objective functions of which is then used as the projection index. The choices include "skew" (skewness), "pow3" (excess kurtosis), "tanh" ($log(cosh)$) and "gauss" (Gaussian function).
alpha	Vector of positive weights between 0 and 1 given to the non-linearities. The length of alpha should be either one less than the number of non-linearities in which case the missing weight is chosen so that alpha sums to one, or equal to the number of non-linearities in which case the weights are used as such. No boundary checks for the weights are done.
N	Number of normal samples to be used in simulating the distribution of the test statistic under the null hypothesis.
eps	Convergence tolerance.
verbose	If TRUE the numbers of iterations will be printed.
maxiter	Maximum number of iterations.

Details

It is assumed that the data is a random sample from the model $x = m + As$ where the latent vector $s = (s_1^T, s_2^T)^T$ consists of k-dimensional non-Gaussian subvector (the signal) and $p-k$-dimensional Gaussian subvector (the noise) and the components of s are mutually independent. Without loss of generality we further assume that the components of s have zero means and unit variances.

The algorithm first estimates full p components from the data using deflation-based NGPP with the chosen non-linearities and weighting and then tests the null hypothesis $H_0 : k_{true} \leq k$ for each $k = 0, \ldots, p-1$. The testing is based on the fact that under the null hypothesis $H_0 : k_{true} \leq k$ the distribution of the final $p - k$ components is standard multivariate normal and the significance of the test can be obtained by comparing the objective function value of the $(k+1)$th estimated

components to the same quantity estimated from N samples of size n from $(p-k)$-dimensional standard multivariate normal distribution.

Note that if maxiter is reached at any step of the algorithm it will use the current estimated direction and continue to the next step.

Value

A list with class 'icest' containing the following components:

statistic	Test statistic, i.e. the objective function values of all estimated component.
p.value	Obtained vector of p-values.
parameter	Number N of simulated normal samples.
method	Character string "Estimation the signal subspace dimension using NGPP".
data.name	Character string giving the name of the data.
W	Estimated unmixing matrix
S	Matrix of size $n \times p$ containing the estimated signals.
D	Vector of the objective function values of the signals
MU	Location vector of the data which was substracted before estimating the signal components.
conv	Boolean vector telling for which components the algorithm converged (TRUE) and for which not (FALSE).

Author(s)

Joni Virta

References

Virta, J., Nordhausen, K. and Oja, H., (2016), Projection Pursuit for non-Gaussian Independent Components, <https://arxiv.org/abs/1612.05445>.

See Also

NGPP, NGPPsim

Examples

```
# Iris data

X <- as.matrix(iris[, 1:4])

# The number of simulations N should be increased in practical situations
# Now we settle for N = 100

res <- NGPPest(X, N = 100)
res$statistic
res$p.value
res$conv
```

NGPPsim *Signal Subspace Dimension Testing Using non-Gaussian Projection Pursuit*

Description

Tests whether the true dimension of the signal subspace is less than or equal to a given k. The test statistic is a multivariate extension of the classical Jarque-Bera statistic and the distribution of it under the null hypothesis is obtained by simulation.

Usage

```
NGPPsim(X, k, nl = c("skew", "pow3"), alpha = 0.8, N = 1000, eps = 1e-6,
        verbose = FALSE, maxiter = 100)
```

Arguments

X	Numeric matrix with n rows corresponding to the observations and p columns corresponding to the variables.
k	Number of components to estimate, $1 \leq k \leq p$.
nl	Vector of non-linearities, a convex combination of the corresponding squared objective functions of which is then used as the projection index. The choices include "skew" (skewness), "pow3" (excess kurtosis), "tanh" ($log(cosh)$) and "gauss" (Gaussian function).
alpha	Vector of positive weights between 0 and 1 given to the non-linearities. The length of alpha should be either one less than the number of non-linearities in which case the missing weight is chosen so that alpha sums to one, or equal to the number of non-linearities in which case the weights are used as such. No boundary checks for the weights are done.
N	Number of normal samples to be used in simulating the distribution of the test statistic under the null hypothesis.
eps	Convergence tolerance.
verbose	If TRUE the numbers of iterations will be printed.
maxiter	Maximum number of iterations.

Details

It is assumed that the data is a random sample from the model $x = m + As$ where the latent vector $s = (s_1^T, s_2^T)^T$ consists of k-dimensional non-Gaussian subvector (the signal) and $p-k$-dimensional Gaussian subvector (the noise) and the components of s are mutually independent. Without loss of generality we further assume that the components of s have zero means and unit variances.

To test the null hypothesis $H_0 : k_{true} \leq k$ the algorithm first estimates $k + 1$ components using delfation-based NGPP with the chosen non-linearities and weighting. Under the null hypothesis the distribution of the final $p - k$ components is standard multivariate normal and the significance of the test is obtained by comparing the objective function value of the $(k + 1)$th estimated components to

the same quantity estimated from N samples of size n from $(p-k)$-dimensional standard multivariate normal distribution.

Note that if maxiter is reached at any step of the algorithm it will use the current estimated direction and continue to the next step.

Value

A list with class 'ictest', inheriting from the class 'hctest', containing the following components:

statistic	Test statistic, i.e. the objective function value of the (k + 1)th estimated component.
p.value	Obtained p-value.
parameter	Number N of simulated normal samples.
method	Character string denoting which test was performed.
data.name	Character string giving the name of the data.
alternative	Alternative hypothesis, i.e. "There are less than p -k Gaussian components".
k	Tested dimension k.
W	Estimated unmixing matrix
S	Matrix of size $n \times (k+1)$ containing the estimated signals.
D	Vector of the objective function values of the signals
MU	Location vector of the data which was substracted before estimating the signal components.

Author(s)

Joni Virta

References

Virta, J., Nordhausen, K. and Oja, H., (2016), Projection Pursuit for non-Gaussian Independent Components, <https://arxiv.org/abs/1612.05445>.

See Also

NGPP, NGPPest

Examples

```
# Simulated data with 2 signals and 2 noise components

n <- 200
S <- cbind(rexp(n), rbeta(n, 1, 2), rnorm(n), rnorm(n))
A <- matrix(rnorm(16), ncol = 4)
X <- S %*% t(A)

# The number of simulations N should be increased in practical situations
# Now we settle for N = 100
```

```
res1 <- NGPPsim(X, 1, N = 100)
res1
screeplot(res1)

res2 <- NGPPsim(X, 2, N = 100)
res2
screeplot(res2)
```

PCAasymp *Testing for Subsphericity using the Covariance Matrix or Tyler's Shape Matrix*

Description

The function tests, assuming an elliptical model, that the last p-k eigenvalues of a scatter matrix are equal and the k interesting components are those with a larger variance. The scatter matrices that can be used here are the regular covariance matrix and Tyler's shape matrix.

Usage

```
PCAasymp(X, k, scatter = "cov", ...)
```

Arguments

X	a numeric data matrix with p>1 columns.
k	the number of eigenvalues larger than the equal ones. Can be between 0 and p-2.
scatter	the scatter matrix to be used. Can be "cov" or "tyler". For "cov" the regular covariance matrix is computed and for "tyler" the function HR.Mest is used to compute Tyler's shape matrix.
...	arguments passed on to HR.Mest if scatter = "tyler".

Details

The functions assumes an elliptical model and tests if the last $p - k$ eigenvalues of PCA are equal. PCA can here be either be based on the regular covariance matrix or on Tyler's shape matrix.

For a sample of size n, the test statistic is

$$T = n/(2\bar{d}^2\sigma_1)\sum_{k+1}^{p}(d_i - \bar{d})^2,$$

where $\bar{d}$ is the mean of the last $p - k$ PCA eigenvalues.

The constant σ_1 is for the regular covariance matrix estimated from the data whereas for Tyler's shape matrix it is simply a function of the dimension of the data.

The test statistic has a limiting chisquare distribution with $(p - k - 1)(p - k + 2)/2$ degrees of freedom.

Note that the regular covariance matrix is here divided by n and not by $n - 1$.

Value

A list of class ictest inheriting from class htest containing:

`statistic`	the value of the test statistic.
`p.value`	the p-value of the test.
`parameter`	the degrees of freedom of the test.
`method`	character string which test was performed.
`data.name`	character string giving the name of the data.
`alternative`	character string specifying the alternative hypothesis.
`k`	the number or larger eigenvalues used in the testing problem.
`W`	the transformation matrix to the principal components.
`S`	data matrix with the centered principal components.
`D`	the underlying eigenvalues.
`MU`	the location of the data which was substracted before calculating the principal components.
`SCATTER`	the computed scatter matrix.
`sigma1`	the asymptotic constant needed for the asymptotic test.

Author(s)

Klaus Nordhausen

References

Nordhausen, K., Oja, H. and Tyler, D.E. (2016), Asymptotic and bootstrap tests for subspace dimension, <https://arxiv.org/abs/1611.04908>.

See Also

`HR.Mest`, `PCAboot`

Examples

```
n <- 200
X <- cbind(rnorm(n, sd = 2), rnorm(n, sd = 1.5), rnorm(n), rnorm(n), rnorm(n))

TestCov <- PCAasymp(X, k = 2)
TestCov
TestTyler <- PCAasymp(X, k = 1, scatter = "tyler")
TestTyler
```

PCAboot *Bootstrap-Based Testing for Subsphericity*

Description

The function tests, assuming an elliptical model, that the last p-k eigenvalues of a scatter matrix are equal and the k interesting components are those with a larger variance. To obtain p-values two different bootstrapping strategies are available and the user can provide the scatter matrix to be used as a function.

Usage

```
PCAboot(X, k, n.boot = 200, s.boot = "B1", S = MeanCov, Sargs = NULL)
```

Arguments

X	a numeric data matrix with p>1 columns.
k	the number of eigenvalues larger than the equal ones. Can be between 0 and p-2.
n.boot	number of bootstrapping samples.
s.boot	bootstrapping strategy to be used. Possible values are "B1", "B2". See details for further information.
S	A function which returns a list that has as its first element a location vector and as the second element the scatter matrix.
Sargs	list of further arguments passed on to the function specified in S.

Details

Here the function S needs to return a list where the first argument is a location vector and the second one a scatter matrix.

The location is used to center the data and the scatter matrix is used to perform PCA.

Consider X as the centered data and denote by W the transformation matrix to the principal components. The corresponding eigenvalues from PCA are $d_1, ..., d_p$. Under the null, $d_k > d_{k+1} = ... = d_p$. Denote further by $\bar{d}$ the mean of the last p-k eigenvalues and by $D^* = diag(d_1, ..., d_k, \bar{d}, ..., \bar{d})$ a $p \times p$ diagonal matrix. Assume that S is the matrix with principal components which can be decomposed into S_1 and S_2 where S_1 contains the k interesting principal components and S_2 the last $p - k$ principal components.

For a sample of size n, the test statistic used for the boostrapping tests is

$$T = n/(\bar{d}^2) \sum_{k+1}^{p} (d_i - \bar{d})^2.$$

The function offers then two boostrapping strategies:

1. s.boot="B1": The first strategy has the following steps:

(a) Take a bootstrap sample S^* of size n from S and decompose it into S_1^* and S_2^*.
(b) Every observation in S_2^* is transformed with a different random orthogonal matrix.
(c) Recombine $S^* = (S_1^*, S_2^*)$ and create $X^* = S^*W$.
(d) Compute the test statistic based on X^*.
(e) Repeat the previous steps `n.boot` times.

2. `s.boot="B2"`: The second strategy has the following steps:
(a) Scale each principal component using the matrix D, i.e. $Z = SD$.
(b) Take a bootstrap sample Z^* of size n from Z.
(c) Every observation in Z^* is transformed with a different random orthogonal matrix.
(d) Recreate $X^* = Z^* D^{*-1} W$.
(e) Compute the test statistic based on X^*.
(f) Repeat the previous steps `n.boot` times.

To create the random orthogonal matrices the function `rorth` is used.

Value

A list of class ictest inheriting from class htest containing:

`statistic`	the value of the test statistic.
`p.value`	the p-value of the test.
`parameter`	the degrees of freedom of the test.
`method`	character string which test was performed.
`data.name`	character string giving the name of the data.
`alternative`	character string specifying the alternative hypothesis.
`k`	the number or larger eigenvalues used in the testing problem.
`W`	the transformation matrix to the principal components.
`S`	data matrix with the centered principal components.
`D`	the underlying eigenvalues.
`MU`	the location of the data which was substracted before calculating the principal components.
`SCATTER`	The computed scatter matrix.
`scatter`	character string denoting which scatter function was used.
`s.boot`	character string denoting which bootstrapping test version was used.

Author(s)

Klaus Nordhausen

References

Nordhausen, K., Oja, H. and Tyler, D.E. (2016), Asymptotic and bootstrap tests for subspace dimension, <https://arxiv.org/abs/1611.04908>.

See Also

cov, MeanCov, PCAasymp

Examples

```
n <- 200
X <- cbind(rnorm(n, sd = 2), rnorm(n, sd = 1.5), rnorm(n), rnorm(n), rnorm(n))

# for demonstration purpose the n.boot is chosen small, should be larger in real applications

TestCov <- PCAboot(X, k = 2, n.boot=30)
TestCov


TestTM <- PCAboot(X, k = 1, n.boot=30, s.boot = "B2", S = "tM", Sargs = list(df=2))
TestTM
```

PCAladle *Ladle Estimate for PCA*

Description

For p-variate data, the Ladle estimate for PCA assumes that the last p-k eigenvalues are equal. Combining information from the eigenvalues and eigenvectors of the covariance matrix the ladle estimator yields an estimate for k.

Usage

```
PCAladle(X, n.boot = 200,
        ncomp = ifelse(ncol(X) > 10, floor(ncol(X)/log(ncol(X))), ncol(X) - 1))
```

Arguments

X	numeric data matrix.
n.boot	number of bootstrapping samples to be used.
ncomp	The number of components among which the ladle estimator is to be searched. The default here follows the recommendation of Luo and Li 2016.

Details

The model here assumes that the eigenvalues of the covariance matrix are of the form $\lambda_1 \geq ... \geq \lambda_k > \lambda_{k+1} = ... = \lambda_p$ and the goal is to estimate the value of k. The ladle estimate for this purpose combines the values of the scaled eigenvalues and the variation of the eigenvectors based on bootstrapping. The idea there is that for distinct eigenvales the variation of the eigenvectors is small and for equal eigenvalues the corresponding eigenvectors have large variation.

This measure is then computed assuming k=0,..., ncomp and the ladle estimate for k is the value where the measure takes its minimum.

Value

A list of class ladle containing:

`method`	the string PCA.
`k`	the estimated value of k.
`fn`	vector giving the measures of variation of the eigenvectors using the bootstrapped eigenvectors for the different number of components.
`phin`	normalized eigenvalues of the covariance matrix.
`gn`	the main criterion for the ladle estimate - the sum of fn and phin. k is the value where gn takes its minimum
`lambda`	the eigenvalues of the covariance matrix.
`W`	the transformation matrix to the principal components.
`S`	data matrix with the centered principal components.
`MU`	the location of the data which was substracted before calculating the principal components.
`data.name`	the name of the data for which the ladle estimate was computed.

Author(s)

Klaus Nordhausen

References

Luo, W. and Li, B. (2016), Combining Eigenvalues and Variation of Eigenvectors for Order Determination, Biometrika, 103, 875–887. <doi:10.1093/biomet/asw051>

See Also

`ladleplot`

Examples

```
n <- 1000
Y <- cbind(rnorm(n, sd=2), rnorm(n,sd=2), rnorm(n), rnorm(n), rnorm(n), rnorm(n))

testPCA <- PCAladle(Y)
testPCA
summary(testPCA)
plot(testPCA)
ladleplot(testPCA)
ladleplot(testPCA, crit = "fn")
ladleplot(testPCA, crit = "lambda")
ladleplot(testPCA, crit = "phin")
```

PCAschott *Testing for Subsphericity using the Schott's test*

Description

The test tests the equality of the last eigenvalues assuming normal distributed data using the regular covariance matrix.

Usage

```
PCAschott(X, k)
```

Arguments

X	a numeric data matrix with p>1 columns.
k	the number of eigenvalues larger than the equal ones. Can be between 0 and p-2.

Details

The functions assumes multivariate normal data and tests if the last $p - k$ eigenvalues of PCA are equal.

Value

A list of class ictest inheriting from class htest containing:

statistic	the value of the test statistic.
p.value	the p-value of the test.
parameter	the degrees of freedom of the test.
method	character string which test was performed.
data.name	character string giving the name of the data.
alternative	character string specifying the alternative hypothesis.
k	the number or larger eigenvalues used in the testing problem.
W	the transformation matrix to the principal components.
S	data matrix with the centered principal components.
D	the underlying eigenvalues.
MU	the mean vector of the data which was substracted before calculating the principal components.
SCATTER	the computed covariance matrix matrix.

Author(s)

Klaus Nordhausen

References

Schott, J.R. (2006), A High-Dimensional Test for the Equality of the Smallest Eigenvalues of a Covariance Matrix, Journal of Multivariate Analysis, 97, 827–843. <doi:10.1016/j.jmva.2005.05.003>

See Also

PCAasymp, PCAboot

Examples

```
n <- 200
X <- cbind(rnorm(n, sd = 2), rnorm(n, sd = 1.5), rnorm(n), rnorm(n), rnorm(n))
PCAschott(X, 2)
```

---

plot.ictest *Scatterplot Matrix for a ictest Object*

Description

For an object of class ictest, plots either the pairwise scatter plot matrix, or the time series plots of the underlying components. The user can choose if only the components considered interesting or all of them should be plotted.

Usage

```
## S3 method for class 'ictest'
plot(x, which = "all", ...)
```

Arguments

x	object of class ictest
which	if "all", then all components of S in the ictest object are plotted. If "k", then only the first k components are plotted, where the value of k is taken from the ictest object. This is only meaningful if k was at least 2.
...	other arguments passed on to pairs if the components are a numeric matrix or to plot.ts, plot.zoo or plot.xts if the components are from the corresponding class.

Details

If the component matrix has the class mts, xts or zoo, then a time series plot will be plotted. Otherwise, the pairwise scatter plot matrix will be plotted.

Author(s)

Klaus Nordhausen

See Also

ggplot.ictest, pairs, plot.ts, plot.zoo, plot.xts

Examples

```
n <- 200
X <- cbind(rnorm(n, sd = 2), rnorm(n, sd = 1.5), rnorm(n), rnorm(n), rnorm(n))

TestCov <- PCAasymp(X, k = 2)
plot(TestCov)
plot(TestCov, which = "k")
```

---

plot.ladle *Plotting an Object of Class ladle*

Description

An object of class ladle contains always the source components as estimated by the corresponding statistical method. This function either plots all of the components or only this considered interesting according to the ladle estimate.

Usage

```
## S3 method for class 'ladle'
plot(x, which = "all", ...)
```

Arguments

x	an object of class ladle.
which	if "all", then all components of S in the ladle object are plotted. If "k", then only the k components are plotted, which are considered interesting according to the ladle estimator. This is only meaningful if the estimated 'k' is at least 2.
...	other arguments passed on to pairs if the components are a numeric matrix or to plot.ts, plot.zoo or plot.xts if the components are from the corresponding class.

Details

If the component matrix has the class mts, xts or zoo, then a time series plot will be plotted. Otherwise, the pairwise scatter plot matrix will be plotted.

Author(s)

Klaus Nordhausen

See Also

pairs, plot.ts, plot.zoo, plot.xts

Examples

```
n <- 1000
X <- cbind(rexp(n), rt(n,5), rnorm(n), rnorm(n), rnorm(n), rnorm(n))
test <- FOBIladle(X)
plot(test)
```

`print.ladle` *Printing an Object of Class ladle*

Description

Basic printing of an object of class ladle. Prints basically everything but the estimated components.

Usage

```
## S3 method for class 'ladle'
print(x, ...)
```

Arguments

`x`	an object of class ladle.
`...`	further arguments to be passed to or from methods.

Author(s)

Klaus Nordhausen

See Also

`summary.ladle`, `plot.ladle`, `ladleplot`, `FOBIladle`, `PCAladle`, `SIRladle`

Examples

```
n <- 1000
X <- cbind(rexp(n), rt(n,5), rnorm(n), rnorm(n), rnorm(n), rnorm(n))
test <- FOBIladle(X)
test
```

rMU *Greek Letter mu Shaped Bivariate Data Generation*

Description

A function to generate bivariate data with the scatterplot resembling the greek letter mu.

Usage

```
rMU(n)
```

Arguments

n	the sample size.

Value

A n times 2 matrix

Author(s)

Klaus Nordhausen, Joni Virta

Examples

```
x <- rMU(1000)

plot(x)
```

rOMEGA *Greek Letter Omega Shaped Bivariate Data Generation*

Description

A function to generate bivariate data with the scatterplot resembling the greek letter Omega.

Usage

```
rOMEGA(n)
```

Arguments

n	the sample size.

Value

A n times 2 matrix

Author(s)

Klaus Nordhausen, Joni Virta

Examples

```
x <- rOMEGA(1000)

plot(x)
```

`rorth` *Random Orthogonal Matrix Creation Uniform WRT the Haar Measure.*

Description

A function to create a random orthogonal matrix uniformly distributed with respect to the Haar measure.

Usage

```
rorth(k)
```

Arguments

k	the desired numer of columns (=rows) of the orthogonal matrix.

Details

The function fills a kxk matrix with N(0,1) random variables and perfroms then a QR decompoistion using `qr`. If the diagonal elements of R are all positive the resulting orthogonal matrix Q is uniform distributed wrt to the Haar measure. Note that the function currently does not check if all diagonal measurements are indeed positive (however this will happen with probability 1 in theory).

Value

An orthogonal k times k matrix

Author(s)

Klaus Nordhausen

References

Stewart, G.W. (1980), The efficient generation of random orthogonal matrices with an application to condition estimators, SIAM Journal on Numerical Analysis, ***17***, *403–409. <doi:10.1137/0717034>.*

Examples

```
Orth <- rorth(4)

crossprod(Orth)
tcrossprod(Orth)
```

`screeplot.ictest` *Screeplot for an ictest Object*

Description

Plots the criterion values of an `ictest` object against its index number. Two versions of this screeplot are available.

Usage

```
## S3 method for class 'ictest'
screeplot(x, type = "barplot", main = deparse(substitute(x)),
  ylab = "criterion", xlab = "component", ...)
```

Arguments

`x`	object of class `ictest`.
`type`	`barplot` if a barplot or `lines` if a line plot is preferred.
`main`	main title of the plot.
`ylab`	y-axis label.
`xlab`	x-axis label.
`...`	other arguments for the plotting functions.

Author(s)

Klaus Nordhausen

See Also

ggscreeplot

Examples

```
n <- 200
X <- cbind(rnorm(n, sd = 2), rnorm(n, sd = 1.5), rnorm(n), rnorm(n), rnorm(n))

TestCov <- PCAasymp(X, k = 2)
screeplot(TestCov)
```

SIRasymp *Testing the Subspace Dimension for Sliced Inverse Regression.*

Description

Using the two scatter matrices approach (SICS) for sliced inversion regression (SIR), the function tests if the last p-k components have zero eigenvalues, where p is the number of explaining variables. Hence the assumption is that the first k components are relevant for modelling the response y and the remaining components are not.

Usage

```
SIRasymp(X, y, k, h = 10, ...)
```

Arguments

X	a numeric data matrix of explaining variables.
y	a numeric vector specifying the response.
k	the number of relevant components under the null hypothesis.
h	the number of slices used in SIR. Passed on to function covSIR.
...	other arguments passed on to covSIR.

Details

Under the null the first k eigenvalues contained in D are non-zero and the remaining p-k are zero.

For a sample of size n, the test statistic T is then n times the sum of these last p-k eigenvalue and has under the null a chisquare distribution with $(p-k)(h-k-1)$ degrees of freedom, therefore it is required that $k < h - 1$.

Value

A list of class ictest inheriting from class htest containing:

statistic	the value of the test statistic.
p.value	the p-value of the test.
parameter	the degrees of freedom of the test.
method	character string which test was performed.
data.name	character string giving the name of the data.
alternative	character string specifying the alternative hypothesis.
k	the number of non-zero eigenvalues used in the testing problem.
W	the transformation matrix to the underlying components.
S	data matrix with the centered underlying components.
D	the underlying eigenvalues.
MU	the location of the data which was substracted before calculating the components.

Author(s)

Klaus Nordhausen

References

Nordhausen, K., Oja, H. and Tyler, D.E. (2016), Asymptotic and bootstrap tests for subspace dimension, <https://arxiv.org/abs/1611.04908>.

See Also

covSIR, SIRboot

Examples

```
X <- matrix(rnorm(1000), ncol = 5)
eps <- rnorm(200, sd = 0.1)
y <- 2 + 0.5 * X[, 1] + 2 * X[, 3] + eps

SIRasymp(X, y, k = 0)
SIRasymp(X, y, k = 1)
```

SIRboot — *Testing the Subspace Dimension for Sliced Inverse Regression Using Bootstrapping.*

Description

Using the two scatter matrices approach (SICS) for sliced inversion regression (SIR) the function tests if the last p-k components have zero eigenvalues, where p is the number of explaining variables. Hence the assumption is that the first k components are relevant for modelling the response y and the remaining components are not. The function performs bootstrapping to obtain a p-value.

Usage

```
SIRboot(X, y, k, h = 10, n.boot = 200, ...)
```

Arguments

X	a numeric data matrix of explaining variables.
y	a numeric vector specifying the response.
k	the number of relevant components under the null hypothesis.
h	the number of slices used in SIR. Passed on to function covSIR.
n.boot	number of bootstrapping samples.
...	other arguments passed on to covSIR.

Details

Under the null hypthesis the last p-k eigenvalue as given in D are zero. The test statistic is then the sum of these eigenvalues.

Denote W as the transformation matrix to the supervised invariant coordinates (SIC) s_i, $i = 1, \ldots, n$, i.e.

$$s_i = W(x_i - MU),$$

where `MU` is the location.

Let S_1 be the submatrix of the SICs which are relevant and S_2 the submatrix of the SICs which are irrelevant for the response y under the null.

The boostrapping has then the following steps:

1. Take a boostrap sample (y^*, S_1^*) of size n from (y, S_1).
2. Take a boostrap sample S_2^* of size n from S_2.
3. Combine $S^* = (S_1^*, S_2^*)$ and create $X^* = S^*W$.
4. Compute the test statistic based on X^*.
5. Repeat the previous steps `n.boot` times.

Value

A list of class ictest inheriting from class htest containing:

`statistic`	the value of the test statistic.
`p.value`	the p-value of the test.
`parameter`	the number of boostrapping samples used to compute the p-value.
`method`	character string which test was performed.
`data.name`	character string giving the name of the data.
`alternative`	character string specifying the alternative hypothesis.
`k`	the number of non-zero eigenvalues used in the testing problem.
`W`	the transformation matrix to the underlying components.
`S`	data matrix with the centered underlying components.
`D`	the underlying eigenvalues.
`MU`	the location of the data which was substracted before calculating the components.

Author(s)

Klaus Nordhausen

References

Nordhausen, K., Oja, H. and Tyler, D.E. (2016), Asymptotic and bootstrap tests for subspace dimension, <https://arxiv.org/abs/1611.04908>.

See Also

covSIR, SIRasymp

Examples

```
X <- matrix(rnorm(1000), ncol = 5)
eps <- rnorm(200, sd = 0.1)
y <- 2 + 0.5 * X[, 1] + 2 * X[, 3] + eps

SIRboot(X, y, k = 0)
SIRboot(X, y, k = 1)
```

SIRladle *Ladle Estimate for SIR*

Description

In the supervised dimension reduction context with response y and explaining variables x, this functions provides the ladle estimate for the dimension of the central subspace for SIR.

Usage

```
SIRladle(X, y, h = 10, n.boot = 200,
      ncomp = ifelse(ncol(X) > 10, floor(ncol(X)/log(ncol(X))), ncol(X) - 1), ...)
```

Arguments

X	numeric data matrix.
y	numeric response vector.
h	number of slices in SIR.
n.boot	number of bootstrapping samples to be used.
ncomp	The number of components among which the ladle estimator is to be searched. The default here follows the recommendation of Luo and Li 2016.
...	arguments passed on to quantile.

Details

The idea here is that the eigenvalues of the SIR-M matrix are of the form $\lambda_1 \geq ... \geq \lambda_k > 0 = ... = 0$ and the eigenvectors of the non-zero eigenvalue span the central subspace.

The ladle estimate for k for this purpose combines the values of the scaled eigenvalues and the variation of the eigenvectors based on bootstrapping. The idea there is that for distinct eigenvales the variation of the eigenvectors is small and for equal eigenvalues the corresponding eigenvectors have large variation.

This measure is then computed assuming k=0,..., ncomp and the ladle estimate for k is the value where the measure takes its minimum.

Value

A list of class ladle containing:

method	the string SIR.
k	the estimated value of k.
fn	vector giving the measures of variation of the eigenvectors using the bootstrapped eigenvectors for the different number of components.
phin	normalized eigenvalues of the M matrix in the SIR case.
gn	the main criterion for the ladle estimate - the sum of fn and phin. k is the value where gn takes its minimum
lambda	the eigenvalues of the M matrix in the SIR case.
W	the transformation matrix to supervised components.
S	data matrix with the centered supervised components.
MU	the location of the data which was substracted before calculating the supervised components.
data.name	the name of the data for which the ladle estimate was computed.

Author(s)

Klaus Nordhausen

References

Luo, W. and Li, B. (2016), Combining Eigenvalues and Variation of Eigenvectors for Order Determination, Biometrika, 103. 875–887. <doi:10.1093/biomet/asw051>

See Also

ladleplot

Examples

```
n <- 1000
X <- cbind(rnorm(n), rnorm(n), rnorm(n), rnorm(n), rnorm(n))
eps <- rnorm(n, sd=0.02)
y <- 4*X[,1] + 2*X[,2] + eps


test <- SIRladle(X, y)
test
summary(test)
plot(test)
pairs(cbind(y, components(test)))
ladleplot(test)
ladleplot(test, crit = "fn")
ladleplot(test, crit = "lambda")
ladleplot(test, crit = "phin")
```

summary.ladle *Summarizing an Object of Class ladle*

Description

Summarizes an ladle object

Usage

```
## S3 method for class 'ladle'
summary(object, ...)
```

Arguments

object	an object of class ladle.
...	further arguments to be passed to or from methods.

Author(s)

Klaus Nordhausen

See Also

print.ladle, plot.ladle, ladleplot, FOBIladle, PCAladle, SIRladle

Examples

```
n <- 1000
X <- cbind(rexp(n), rt(n,5), rnorm(n), rnorm(n), rnorm(n), rnorm(n))

test <- FOBIladle(X)
summary(test)
```

Index