Package ‘ICSOutlier’

February 3, 2018

Type Package

Title Outlier Detection Using Invariant Coordinate Selection

Version 0.3-0

Date 2018-02-03

Author Aurore Archimbaud, Klaus Nordhausen, Anne Ruiz-Gazen

Maintainer Klaus Nordhausen <klaus.nordhausen@tuwien.ac.at>

Depends R (>= 3.0.0), methods, ICS (>= 1.3-0), moments

Imports graphics, grDevices, mvtnorm, parallel

Suggests REPPlab

Description Multivariate outlier detection is performed using invariant coordinates where the package offers different methods to choose the appropriate components.

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2018-02-02 23:07:18 UTC

R topics documented:

comp.norm.test *Selection of Nonnormal Invariant Components Using Marginal Normality Tests*

Description

Identifies invariant coordinates that are non normal using univariate normality tests.

Usage

```
comp.norm.test(object, test = "agostino.test", type = "smallprop", level = 0.05,
  adjust = TRUE)
```

Arguments

`object`	object of class `ics2` where both `S1` and `S2` are specified as functions. The sample size and the dimension of interest are also obtained from the object.
`test`	name of the normality test to be used. Possibilites are `"jarque.test"`, `"anscombe.test"`, `"bonett.test"`, `"agostino.test"`, `"shapiro.test"`. Default is `"agostino.test"`.
`type`	currently the only option is `"smallprop"`. See details.
`level`	the initial level used to make a decision based on the test p-values. See details.
`adjust`	logical. If `TRUE`, the quantiles levels are adjusted. Default is `TRUE`. See details.

Details

Currently the only available `type` is `"smallprop"` which detects which of the components follow a univariately normal distribution. It starts from the first component and stops when a component is detected as gaussian. Five tests for univariate normality are available.

If `adjust = FALSE` all tests are performed at the same `level`. This leads however often to too many components. Therefore some multiple testing adjustments might be useful. The current default adjusts the level for the jth component as `level`/j.

Note that the function is seldomly called directly by the user but internally by `ics.outlier`.

Value

A list containing:

`index`	integer vector indicating the indices of the selected components.
`test`	string with the name of the normality test used.
`criterion`	vector of the p-values from the marginal normality tests for each component.
`levels`	vector of the levels used for the decision for each component.
`adjust`	logical. `TRUE` if adjusted.
`type`	`type` used.

Author(s)

Aurore Archimbaud and Klaus Nordhausen

References

Archimbaud, A., Nordhausen, K. and Ruiz-Gazen, A. (2016), Multivariate Outlier Detection With ICS, <https://arxiv.org/abs/1612.06118>.

See Also

`ics2`, `comp.simu.test`, `jarque.test`, `anscombe.test`, `bonett.test`, `agostino.test`, `shapiro.test`

Examples

```
Z <- rmvnorm(1000, rep(0, 6))
# Add 20 outliers on the first component
Z[1:20, 1] <- Z[1:20, 1] + 10
pairs(Z)
icsZ <- ics2(Z)
# The shift located outliers can be displayed in one dimension
comp.norm.test(icsZ)
# Only one invariant component is non normal and selected.
comp.norm.test(icsZ, test = "bo")

# Example with no outlier
Z0 <- rmvnorm(1000, rep(0, 6))
pairs(Z0)
icsZ0 <- ics2(Z0)
# Should select no component
comp.norm.test(icsZ0, level = 0.01)$index
```

---

comp.simu.test *Selection of Nonnormal Invariant Components Using Simulations*

Description

Identifies invariant coordinates that are nonnormal using simulations under a standard multivariate normal model for a specific data setup and scatter combination.

Usage

```
comp.simu.test(object, m = 10000, type = "smallprop", level = 0.05,
  adjust = TRUE, ncores = NULL, iseed = NULL, pkg = "ICSOutlier",
  qtype = 7, ...)
```

Arguments

object	object of class ics2 where both S1 and S2 are specified as functions. The sample size and the dimension of interest are also obtained from the object.
m	number of simulations. Note that since extreme quantiles are of interest m should be large.
type	currently the only type option is "smallprop". See details.
level	the initial level used to make a decision. The cut-off values are the (1-level)th quantile of the eigenvalues obtained from simulations. See details.
adjust	logical. If TRUE, the quantiles levels are adjusted. Default is TRUE. See details.
ncores	number of cores to be used. If NULL or 1, no parallel computing is used. Otherwise makeCluster with type = "PSOCK" is used.
iseed	If parallel computation is used the seed passed on to clusterSetRNGStream. Default is NULL which means no fixed seed is used.
pkg	When using parallel computing, a character vector listing all the packages which need to be loaded on the different cores via require. Must be at least "ICSOutlier" and must contain the packages needed to compute the scatter matrices.
qtype	specifies the quantile algorithm used in quantile.
...	further arguments passed on to the function quantile.

Details

Based on simulations it detects which of the components follow a univariately normal distribution. More precisely it identifies the observed eigenvalues larger than the ones coming from normal distributed data. m standard normal data sets are simulated using the same data size and scatters as specified in the ics2 object. The cut-off values are determined based on a quantile of these simulated eigenvalues.

As the eigenvalues, aka generalized kurtosis values, of ICS are ordered it is natural to perform the comparison in a specific order depending on the purpose. Currently the only available type is "smallprop" so starting with the first component, the observed eigenvalues are successively compared to these cut-off values. The precedure stops when an eigenvalue is below the corresponding cut-off, so when a normal component is detected.

If adjust = FALSE all eigenvalues are compared to the same (1-level)th level of the quantile. This leads however often to too many selected components. Therefore some multiple testing adjustment might be useful. The current default adjusts the quantile for the jth component as 1-level/j.

Note that depending on the data size and scatters used this can take a while and so it is more efficient to parallelize computations. Note also that the function is seldomly called directly by the user but internally by ics.outlier.

Value

A list containing:

index	integer vector indicating the indices of the selected components.
test	string "simulation".

`criterion`	vector of the cut-off values for all the eigenvalues.
`levels`	vector of the levels used to derive the cut-offs for each component.
`adjust`	logical. `TRUE` if adjusted.
`type`	`type` used.
`m`	number of iterations `m` used in the simulations.

Author(s)

Aurore Archimbaud and Klaus Nordhausen

References

Archimbaud, A., Nordhausen, K. and Ruiz-Gazen, A. (2016), Multivariate Outlier Detection With ICS, <https://arxiv.org/abs/1612.06118>.

See Also

`ics2`, `comp.norm.test`

Examples

```
# For a real analysis use larger values for m and more cores if available

set.seed(123)
Z <- rmvnorm(1000, rep(0, 6))
# Add 20 outliers on the first component
Z[1:20, 1] <- Z[1:20, 1] + 10
pairs(Z)
icsZ <- ics2(Z)
# For demo purpose only small m value, should select the first component
comp.simu.test(icsZ, m = 400, ncores = 1)

## Not run:
# For using two cores
# For demo purpose only small m value, should select the first component
comp.simu.test(icsZ, m = 500, ncores = 2, iseed = 123)

# For using several cores and for using a scatter function from a different package
# Using the parallel package to detect automatically the number of cores
library(parallel)
# ICS with MCD estimates and the usual estimates
# Need to create a wrapper for the CovMcd function to return first the location estimate
# and the scatter estimate secondly.
library(rrcov)
myMCD <- function(x,...){
  mcd <- CovMcd(x,...)
  return(list(location = mcd@center, scatter = mcd@cov))
}
icsZmcd <- ics2(Z, S1 = myMCD, S2 = MeanCov, S1args = list(alpha = 0.75))
# For demo purpose only small m value, should select the first component
comp.simu.test(icsZmcd, m = 500, ncores = detectCores()-1,
```

```
                 pkg = c("ICSOutlier", "rrcov"), iseed = 123)

## End(Not run)

# Example with no outlier
Z0 <- rmvnorm(1000, rep(0, 6))
pairs(Z0)
icsZ0 <- ics2(Z0)
#Should select no component
comp.simu.test(icsZ0, m = 400, level = 0.01, ncores = 1)
```

---

dist.simu.test — *Cut-Off Values Using Simulations for the Detection of Extreme ICS Distances*

Description

Computes the cut-off values for the identification of the outliers based on the squared ICS distances. It uses simulations under a multivariate standard normal model for a specific data setup and scatters combination.

Usage

```
dist.simu.test(object, index, m = 10000, level = 0.025, ncores = NULL,
                iseed = NULL, pkg = "ICSOutlier", qtype = 7, ...)
```

Arguments

`object`	object of class `ics2` where both `S1` and `S2` are specified as functions. The sample size and the dimension of interest are also obtained from the object.
`index`	integer vector specifiying which components are used to compute the `ics.distances`.
`m`	number of simulations. Note that extreme quantiles are of interest and hence `m` should be large.
`level`	the (1-`level`(s))th quantile(s) used to choose the cut-off value(s). Usually just one number between 0 and 1. However a vector is also possible.
`ncores`	number of cores to be used. If `NULL` or 1, no parallel computing is used. Otherwise makeCluster with `type = "PSOCK"` is used.
`iseed`	If parallel computation is used the seed passed on to `clusterSetRNGStream`. Default is NULL which means no fixed seed is used.
`pkg`	When using parallel computing, a character vector listing all the packages which need to be loaded on the different cores via `require`. Must be at least "ICSOutlier" and must contain the packages needed to compute the scatter matrices.
`qtype`	specifies the quantile algorithm used in `quantile`.
`...`	further arguments passed on to the function `quantile`.

Details

The function extracts basically the dimension of the data from the `ics2` object and simulates `m` times, from a multivariate standard normal distribution, the squared ICS distances with the components specified in `index`. The resulting value is then the mean of the `m` correponding quantiles of these distances at level 1-`level`.

Note that depending on the data size and scatters used this can take a while and so it is more efficient to parallelize computations.

Note that the function is seldomly called directly by the user but internally by `ics.outlier`.

Value

A vector with the values of the (1-`level`)th quantile.

Author(s)

Aurore Archimbaud and Klaus Nordhausen

References

Archimbaud, A., Nordhausen, K. and Ruiz-Gazen, A. (2016), Multivariate Outlier Detection With ICS, <https://arxiv.org/abs/1612.06118>.

See Also

`ics2`, `ics.distances`

Examples

```
# For a real analysis use larger values for m and more cores if available

Z <- rmvnorm(1000, rep(0, 6))
Z[1:20, 1] <- Z[1:20, 1] + 10
A <- matrix(rnorm(36), ncol = 6)
X <- tcrossprod(Z, A)

pairs(X)
icsX <- ics2(X)

icsX.dist.1 <- ics.distances(icsX, index = 1)
CutOff <- dist.simu.test(icsX, 1, m = 500, ncores = 1)

# check if outliers are above the cut-off value
plot(icsX.dist.1, col = rep(2:1, c(20, 980)))
abline(h = CutOff)


library(REPPlab)
data(ReliabilityData)
# The observations 414 and 512 are suspected to be outliers
icsReliability <- ics2(ReliabilityData, S1 = MeanCov, S2 = Mean3Cov4)
```

```
# Choice of the number of components with the screeplot: 2
screeplot(icsReliability)
# Computation of the distances with the first 2 components
ics.dist.scree <- ics.distances(icsReliability, index = 1:2)
# Computation of the cut-off of the distances
CutOff <- dist.simu.test(icsReliability, 1:2, m = 50, level = 0.02, ncores = 1)
# Identification of the outliers based on the cut-off value
plot(ics.dist.scree)
abline(h = CutOff)
outliers <- which(ics.dist.scree >= CutOff)
text(outliers, ics.dist.scree[outliers], outliers, pos = 2, cex = 0.9)

## Not run:
# For using three cores
# For demo purpose only small m value, should select the first component
dist.simu.test(icsReliability, 1:2, m = 500, level = 0.02, ncores = 3, iseed = 123)

# For using several cores and for using a scatter function from a different package
# Using the parallel package to detect automatically the number of cores
library(parallel)
# ICS with Multivariate Median and Tyler's Shape Matrix and the usual estimates
library(ICSNP)
icsReliabilityHRMest <- ics2(ReliabilityData, S1 = HR.Mest, S2 = MeanCov,
                             S1args = list(maxiter = 1000))
# Computation of the cut-off of the distances. For demo purpose only small m value.
dist.simu.test(icsReliabilityHRMest,  1:2, m = 500, level = 0.02, ncores = detectCores()-1,
               pkg = c("ICSOutlier","ICSNP"), iseed = 123)

## End(Not run)
```

HTP *Production Measurements of High-Tech Parts*

Description

The HTP data set contains 902 high-tech parts designed for consumer products characterized by 88 tests. These tests are performed to ensure a high quality of the production. All these 902 parts were considered functional and have been sold. However the two parts 581 and 619 showed defects in use and were returned to the manufacturer by the customer. Therefore these two can be considered as outliers.

Usage

```
data("HTP")
```

Format

A data frame with 902 observations and 88 numeric variables V.1 - V.88.

Source

Anonymized data from a nondisclosed manufacturer.

Examples

```
# HTP data: the observations 581 and 619 are considered as outliers
data(HTP)
outliers <- c(581, 619)
boxplot(HTP)


# Outlier detection using ICS
icsHTP <- ics2(HTP)
# Selection of components based on a Normality Test, for demo purpose only small mDist value,
# but as extreme quantiles are of interest mDist should be much larger.
# Also more cores could be used if available.
icsOutlierDA <- ics.outlier(icsHTP, test = "agostino.test", level.test = 0.05,
                            level.dist = 0.02, mDist = 50, ncores = 1)
icsOutlierDA
summary(icsOutlierDA)
plot(icsOutlierDA)
text(outliers, icsOutlierDA@ics.distances[outliers], outliers, pos = 2, cex = 0.9, col = 2)

## Not run:
# Selection of components based on simulations
# This might take a while to run (around 30 minutes)
icsOutlierPA <- ics.outlier(icsHTP, method = "simulation", level.dist = 0.02,
level.test = 0.05, mEig = 10000, mDist = 10000)
icsOutlierPA
summary(icsOutlierPA)
plot(icsOutlierPA)
text(outliers, icsOutlierPA@ics.distances[outliers], outliers, pos = 2, cex = 0.9, col = 2)

## End(Not run)
```

`ics.distances` *Squared ICS Distances for Invariant Coordinates*

Description

Computes the squared ICS distances, defined as the Euclidian distances of the selected centered components.

Usage

```
ics.distances(object, index = NULL)
```

Arguments

`object`	object of class `ics2` where both `S1` and `S2` are specified as functions.
`index`	vector of integers indicating the indices of the components to select.

Details

For outlier detection, the squared ICS distances can be used as a measure of outlierness. Denote as Z the invariant coordinates centered with the location estimate specified in `S1` (for details see `ics2`). Let Z_k be the k components of Z selected by `index`, then the ICS distance of the observation i is defined as:

$$ICSD^2(x_i, k) = ||Z_k||^2.$$

Note that if all components are selected, the ICS distances are equivalent to the Mahlanobis distances computed with respect of the first scatter and associated location specified in `S1`.

Value

A numeric vector containing the squared ICS distances.

Author(s)

Aurore Archimbaud and Klaus Nordhausen

References

Archimbaud, A., Nordhausen, K. and Ruiz-Gazen, A. (2016), Multivariate Outlier Detection With ICS, <https://arxiv.org/abs/1612.06118>.

See Also

`ics2`, `mahalanobis`

Examples

```
Z <- rmvnorm(1000, rep(0, 6))
Z[1:20, 1] <- Z[1:20, 1] + 5
A <- matrix(rnorm(36), ncol = 6)
X <- tcrossprod(Z, A)

pairs(X)
icsX <- ics2(X)

icsX.dist.all <- ics.distances(icsX, index = 1:6)
maha <- mahalanobis(X, center = colMeans(X), cov = cov(X))
# in this case the distances should be the same
plot(icsX.dist.all, maha)
all.equal(icsX.dist.all, maha)

icsX.dist.first <- ics.distances(icsX, index = 1)
plot(icsX.dist.first)
```

ics.outlier *Outlier Detection Using ICS*

Description

In a multivariate framework outlier(s) are detected using ICS. The function works on an object of class `ics2` and decides automatically about the number of invariant components to use to search for the outliers and the number of outliers detected on these components. Currently the function is restricted to the case of searching outliers only on the first components.

Usage

```
ics.outlier(object, method = "norm.test", test = "agostino.test", mEig = 10000,
  level.test = 0.05, adjust = TRUE, level.dist = 0.025, mDist = 10000,
  type = "smallprop", ncores = NULL, iseed = NULL, pkg = "ICSOutlier",
  qtype = 7, ...)
```

Arguments

`object`	object of class `ics2` where both `S1` and `S2` are specified as functions.
`method`	name of the method used to select the ICS components involved to compute ICS distances. Options are `"norm.test"` and `"simulation"`. Depending on the method either `comp.norm.test` or `comp.simu.test` are used.
`test`	name of the marginal normality test to use if `method = "norm.test"`. Possibilities are `"jarque.test"`, `"anscombe.test"`, `"bonett.test"`, `"agostino.test"`, `"shapiro.test"`. Default is `"agostino.test"`.
`mEig`	number of simulations performed to derive the cut-off values for selecting the ICS components. Only if `method = "simulation"`. See `comp.simu.test` for details.
`level.test`	`level` for the `comp.norm.test` or `comp.simu.test` functions. The inital level for selecting the invariant coordinates.
`adjust`	logical. For selecting the invariant coordinates, the level of the test can be adjusted for each component to deal with multiple testing. See `comp.norm.test` and `comp.simu.test` for details. Default is TRUE.
`level.dist`	`level` for the `dist.simu.test` function. The (1-`level`)th quantile used to determine the cut-off value for the ICS distances.
`mDist`	number of simulations performed to derive the cut-off value for the ICS distances. See `dist.simu.test` for details.
`type`	currently the only option is `"smallprop"` which means that only the first ICS components can be selected. See `comp.norm.test` or `comp.simu.test` for details.

ncores	number of cores to be used in dist.simu.test and comp.simu.test. If NULL or 1, no parallel computing is used. Otherwise makeCluster with type = "PSOCK" is used.
iseed	If parallel computation is used the seed passed on to clusterSetRNGStream. Default is NULL which means no fixed seed is used.
pkg	When using parallel computing, a character vector listing all the packages which need to be loaded on the different cores via require. Must be at least "ICSOutlier" and must contain the packages needed to compute the scatter matrices.
qtype	specifies the quantile algorithm used in quantile.
...	passed on to other methods.

Details

The ICS method has attractive properties for outlier detection in the case of a small proportion of outliers. As for PCA three steps have to be performed: (i) select the components most useful for the detection, (ii) compute distances as outlierness measures for all observation and finally (iii) label outliers using some cut-off value.

This function performs these three steps automatically:

(i) For choosing the components of interest two methods are proposed: "norm.test" based on some marginal normality tests (see details in comp.norm.test) or "simulation" based on a parallel analysis (see details in comp.simu.test). These two approaches lie on the intrinsic property of ICS in case of a small proportion of outliers with the choice of S1 "more robust" than S2, which ensures to find outliers on the first components. Indeed when using S1 = MeanCov and S2 = Mean3Cov4, the Invariant Coordinates are ordered according to their classical Pearson kurtosis values in decreasing order. The information to find the outliers should be then contained in the first k nonnormal directions.

(ii) Then the ICS distances are computed as the Euclidian distances on the selected k centered components Z_k.

(iii) Finally the outliers are identified based on a cut-off derived from simulations. If the distance of an observation exceeds the expectation under the normal model, this observation is labeled as outlier (see details in dist.simu.test).

As a rule of thumb, the percentage of contamination should be limited to 10% in case of a mixture of gaussian distributions and using the default combination of locations and scatters for ICS.

Value

an object of class icsOut

Author(s)

Aurore Archimbaud and Klaus Nordhausen

References

Archimbaud, A., Nordhausen, K. and Ruiz-Gazen, A. (2016), Multivariate Outlier Detection With ICS, <https://arxiv.org/abs/1612.06118>.

See Also

ics2, comp.norm.test, comp.simu.test, dist.simu.test, icsOut-class

Examples

```
# ReliabilityData example: the observations 414 and 512 are suspected to be outliers
library(REPPlab)
data(ReliabilityData)
icsReliabilityData <- ics2(ReliabilityData, S1 = tM, S2 = MeanCov)
# For demo purpose only small mDist value, but as extreme quantiles
# are of interest mDist should be much larger. Also number of cores used
# should be larger if available
icsOutlierDA <- ics.outlier(icsReliabilityData, level.dist = 0.01, mDist = 50, ncores = 1)
icsOutlierDA
summary(icsOutlierDA)
plot(icsOutlierDA)

## Not run:
# For using several cores and for using a scatter function from a different package
# Using the parallel package to detect automatically the number of cores
library(parallel)
# ICS with MCD estimates and the usual estimates
# Need to create a wrapper for the CovMcd function to return first the location estimate
# and the scatter estimate secondly.
data(HTP)
library(rrcov)
myMCD <- function(x,...){
  mcd <- CovMcd(x,...)
  return(list(location = mcd@center, scatter = mcd@cov))
}
icsHTP <- ics2(HTP, S1 = myMCD, S2 = MeanCov, S1args = list(alpha = 0.75))
# For demo purpose only small m value, should select the first seven components
icsOutlier <- ics.outlier(icsHTP, mEig = 50, level.test = 0.05, adjust = TRUE,
                          level.dist = 0.025, mDist = 50,
                          ncores =  detectCores()-1, iseed = 123,
                          pkg = c("ICSOutlier", "rrcov"))
icsOutlier

## End(Not run)

# Exemple of no direction and hence also no outlier
set.seed(123)
X = rmvnorm(500, rep(0, 2), diag(rep(0.1,2)))
icsX <- ics2(X)
icsOutlierJB <- ics.outlier(icsX, test = "jarque", level.dist = 0.01,
level.test = 0.01, mDist = 100, ncores = 1)
summary(icsOutlierJB)
plot(icsOutlierJB)
rm(.Random.seed)
```

```
# Example of no outlier
set.seed(123)
X = matrix(rweibull(1000, 4, 4), 500, 2)
X = apply(X,2, function(x){ifelse(x<5 & x>2, x, runif(sum(!(x<5 & x>2)), 5, 5.5))})
icsX <- ics2(X)
icsOutlierAG <- ics.outlier(icsX, test = "anscombe", level.dist = 0.01,
level.test = 0.05, mDist = 100, ncores = 1)
summary(icsOutlierAG)
plot(icsOutlierAG)
rm(.Random.seed)
```

---

icsOut-class *Class icsOut*

Description

A S4 class to store results from performing outlier detection in an ICS context.

Objects from the Class

Objects can be created by calls of the form `new("icsOut", ...)`. But usually objects are created by the function `ics.outlier`.

Slots

`outliers`: Object of class `"integer"`. A vector containing ones for outliers and zeros for non outliers.

`ics.distances`: Object of class `"numeric"`. Vector giving the squared ICS distances of the observations from the invariant coordinates centered with the location estimate specified in `S1`.

`ics.dist.cutoff`: Object of class `"numeric"`. The cut-off for the distances to decide if an observation is outlying or not.

`level.dist`: Object of class `"numeric"`. The level for deciding upon the cut-off value for the ICS distances.

`level.test`: Object of class `"numeric"`. The inital level for selecting the invariant coordinates.

`method`: Object of class `"character"`. Name of the method used to decide upon the number of ICS components.

`index`: Object of class `"numeric"`. Vector giving the indices of the ICS components selected.

`test`: Object of class `"character"`. The name of the normality test as specified in the function call.

`criterion`: Object of class `"numeric"`. Vector giving the marginal levels for the components selection.

`adjust`: Object of class `"logical"`. Wether the initial level used to decide upon the number of components has been adjusted for multiple testing or not.

`type`: Object of class `"character"`. Currently always the string `"smallprop"`.

`mDist`: Object of class `"integer"`. Number of simulations performed to decide upon the cut-off for the ICS distances.

`mEig`: Object of class `"integer"`. Number of simulations performed for selecting the ICS components based on simulations.

`S1name`: Object of class `"character"`. Name of S1 in the original ics2 object.

`S2name`: Object of class `"character"`. Name of S2 in the original ics2 object.

Methods

For this class the following generic functions are available: `print.icsOut`, `summary.icsOut` and `plot.ics`

Note

In case no extractor function for the slots exists, the component can be extracted the usual way using '@'.

Author(s)

Aurore Archimbaud and Klaus Nordhausen

See Also

`ics.outlier`

`ICSOutlier` *Outlier Detection Using Invariant Coordinate Selection*

Description

Multivariate outlier detection is performed using invariant coordinates and the package offers different methods to choose the appropriate components.

Details

Package:	ICS
Type:	Package
Version:	0.3-0
Date:	2018-02-03
License:	GPL (>= 2)

ICS is a general multivariate technique with many applications in multivariate analysis. ICSOutlier offers a selection of functions for automated detection of outliers in the data based on a fitted ics2 object. The current implementation targets data sets with only a small percentage of outliers but future extensions are under preparation.

Author(s)

Aurore Archimbaud, Klaus Nordhausen, Anne Ruiz-Gazen

Maintainer: Klaus Nordhausen, `<klaus.nordhausen@tuwien.ac.at>`

---

plot.icsOut — *Distances Plot for an icsOut Object*

Description

Distances plot for an icsOut object visualizing the separation of the outliers from the good data points.

Usage

```
## S4 method for signature 'icsOut,missing'
plot(x, pch.out = 16, pch.good = 4, col.out = 1, col.good = grey(0.5),
     col.cut = 1, lwd.cut = 1, lty.cut = 1, xlab = "Observation Number",
     ylab = "ICS distances", ...)
```

Arguments

`x`	object of class `icsOut`.
`pch.out`	ploting symbol for the outliers.
`pch.good`	plotting symbol for the 'good' data points.
`col.out`	color for the outliers.
`col.good`	color for the 'good' data points.
`col.cut`	color for cut-off line.
`lwd.cut`	lwd value for cut-off line.
`lty.cut`	lty value for cut-off line.
`xlab`	default x-axis label.
`ylab`	default y-axis label.
`...`	other arguments for `plot`

Details

For the figure the IC distances are plotted versus their index. The cut-off value for distances is given as a horizontal line and all observations above the line are considered as outliers.

Author(s)

Aurore Archimbaud and Klaus Nordhausen

See Also

`icsOut-class` and `ics.outlier`

Examples

```
Z <- rmvnorm(1000, rep(0, 6))
Z[1:20, 1] <- Z[1:20, 1] + 10
A <- matrix(rnorm(36), ncol = 6)
X <- tcrossprod(Z, A)
icsX <- ics2(X)
# For demonstation purposes mDist is small, should be larger for real data analysis
icsXoutliers <- ics.outlier(icsX, mDist = 500)
plot(icsXoutliers, col.out = 2)
```

`print.icsOut` *Vector of Outlier Indicators*

Description

Short statement about how many components are selected for the outlier detection and how many outliers are detected.

Usage

```
## S4 method for signature 'icsOut'
show(object)
```

Arguments

`object`	object of class `icsOut`.

Author(s)

Aurore Archimbaud and Klaus Nordhausen

See Also

`icsOut-class` and `ics.outlier`

`summary.icsOut` *Summarize a icsOut object*

Description

Summarizes and prints an `icsOut` object in an informative way.

Usage

```
## S4 method for signature 'icsOut'
summary(object, digits = 4)
```

Arguments

`object`	object of class `icsOut`.
`digits`	number of digits for the numeric output.

Author(s)

Aurore Archimbaud and Klaus Nordhausen

See Also

`icsOut-class` and `ics.outlier`

Index