Package ‘rrcov’

August 3, 2020

Version 1.5-5

Date 2020-07-31

VersionNote Released 1.5-2 on 2020-01-16 on CRAN

Title Scalable Robust Estimators with High Breakdown Point

Description Robust Location and Scatter Estimation and Robust
Multivariate Analysis with High Breakdown Point:
principal component analysis (Filzmoser and Todorov (2013), <doi:10.1016/j.ins.2012.10.017>),
linear and quadratic discriminant analysis (Todorov and Pires (2007)),
multivariate tests (Todorov and Filzmoser (2010) <doi:10.1016/j.csda.2009.08.015>),
outlier detection (Todorov et al. (2010) <doi:10.1007/s11634-010-0075-2>).
See also Todorov and Filzmoser (2009) <ISBN-13:978-3838108148>,
Todorov and Filzmoser (2010) <doi:10.18637/jss.v032.i03> and
Boudt et al. (2019) <doi:10.1007/s11222-019-09869-x>.

Maintainer Valentin Todorov <valentin.todorov@chello.at>

Depends R (>= 2.10), robustbase (>= 0.92.1), methods

Imports stats, stats4, mvtnorm, lattice, pcaPP

Suggests grid, MASS

LazyLoad yes

Repository CRAN

NeedsCompilation yes

Author Valentin Todorov [aut, cre] (<https://orcid.org/0000-0003-4215-0245>)

Date/Publication 2020-08-03 15:30:02 UTC

R topics documented:

`Appalachia` *Annual maximum streamflow in central Appalachia*

Description

The data on annual maximum streamflow at 104 gaging stations in the central Appalachia region of the United States contains the sample L-moments ratios (L-CV, L-skewness and L-kurtosis) as used by Hosking and Wallis (1997) to illustrate regional freqency analysis (RFA).

Usage

```
data(Appalachia)
```

Format

A data frame with 104 observations on the following 3 variables:

`L-CV` L-coefficient of variation

`L-skewness` L-coefficient of skewness

`L-kurtosis` L-coefficient of kurtosis

Details

The sample L-moment ratios (L-CV, L-skewness and L-kurtosis) of a site are regarded as a point in three dimensional space.

Source

Hosking, J. R. M. and J. R. Wallis (1997), *Regional Frequency Analysis: An Approach Based on L-moments.* Cambridge University Press, p.175–185

References

Neykov, N.M., Neytchev, P.N., Van Gelder, P.H.A.J.M. and Todorov V. (2007), Robust detection of discordant sites in regional frequency analysis, *Water Resources Research*, 43, W06417, doi:10.1029/2006WR005322

Examples

```
data(Appalachia)

# plot a matrix of scatterplots
pairs(Appalachia,
      main="Appalachia data set",
      pch=21,
      bg=c("red", "green3", "blue"))

mcd<-CovMcd(Appalachia)
mcd
plot(mcd, which="dist", class=TRUE)
plot(mcd, which="dd", class=TRUE)

##  identify the discordant sites using robust distances and compare
##  to the classical ones
mcd <- CovMcd(Appalachia)
rd <- sqrt(getDistance(mcd))
ccov <- CovClassic(Appalachia)
cd <- sqrt(getDistance(ccov))
r.out <- which(rd > sqrt(qchisq(0.975,3)))
c.out <- which(cd > sqrt(qchisq(0.975,3)))
cat("Robust: ", length(r.out), " outliers: ", r.out,"\n")
cat("Classical: ", length(c.out), " outliers: ", c.out,"\n")
```

---

biplot-methods *Biplot for Principal Components (objects of class 'Pca')*

Description

Produces a biplot from an object (derived from) Pca-class.

Usage

```
    ## S4 method for signature 'Pca'
biplot(x, choices=1L:2L, scale=1, ...)
```

Arguments

x	an object of class (derived from) "Pca".
choices	length 2 vector specifying the components to plot. Only the default is a biplot in the strict sense.
scale	The variables are scaled by `lambda ^ scale` and the observations are scaled by `lambda ^ (1-scale)` where `lambda` are the singular values as computed by the Principal Components function. Normally `0 <= scale <= 1`, and a warning will be issued if the specified `scale` is outside this range.
...	optional arguments to be passed to the internal graphical functions.

Side Effects

a plot is produced on the current graphics device.

Methods

biplot `signature(x = Pca)`: Plot a biplot, i.e. represent both the observations and variables of a matrix of multivariate data on the same plot. See also `biplot.princomp`.

References

Gabriel, K. R. (1971). The biplot graphical display of matrices with applications to principal component analysis. *Biometrika*, **58**, 453–467.

See Also

`Pca-class`, `PcaClassic`, `PcaRobust-class`.

Examples

```
require(graphics)
biplot(PcaClassic(USArrests, k=2))
```

---

bus *Automatic vehicle recognition data*

Description

The data set bus (Hettich and Bay, 1999) corresponds to a study in automatic vehicle recognition (see Maronna et al. 2006, page 213, Example 6.3)). This data set from the Turing Institute, Glasgow, Scotland, contains measures of shape features extracted from vehicle silhouettes. The images were acquired by a camera looking downward at the model vehicle from a fixed angle of elevation. Each of the 218 rows corresponds to a view of a bus silhouette, and contains 18 attributes of the image.

Usage

```
data(bus)
```

Format

A data frame with 218 observations on the following 18 variables:

`V1` compactness

`V2` circularity

`V3` distance circularity

`V4` radius ratio

`V5` principal axis aspect ratio

`V6` maximum length aspect ratio

`V7` scatter ratio

`V8` elongatedness

`V9` principal axis rectangularity

`V10` maximum length rectangularity

`V11` scaled variance along major axis

`V12` scaled variance along minor axis

`V13` scaled radius of gyration

`V14` skewness about major axis

`V15` skewness about minor axis

`V16` kurtosis about minor axis

`V17` kurtosis about major axis

`V18` hollows ratio

Source

Hettich, S. and Bay, S.D. (1999), The UCI KDD Archive, Irvine, CA:University of California, Department of Information and Computer Science,

'http://kdd.ics.uci.edu'

References

Maronna, R., Martin, D. and Yohai, V., (2006). Robust Statistics: Theory and Methods. Wiley, New York.

Examples

```
## Reproduce Table 6.3 from Maronna et al. (2006), page 213
data(bus)
bus <- as.matrix(bus)

## calculate MADN for each variable
xmad <- apply(bus, 2, mad)
cat("\nMin, Max of MADN: ", min(xmad), max(xmad), "\n")


## MADN vary between 0 (for variable 9) and 34. Therefore exclude
##  variable 9 and divide the remaining variables by their MADNs.
bus1 <- bus[, -9]
madbus <- apply(bus1, 2, mad)
bus2 <- sweep(bus1, 2, madbus, "/", check.margin = FALSE)

## Compute classical and robust PCA (Spherical/Locantore, Hubert, MCD and OGK)
pca  <- PcaClassic(bus2)
rpca <- PcaLocantore(bus2)
pcaHubert <- PcaHubert(bus2, k=17, kmax=17, mcd=FALSE)
pcamcd <- PcaCov(bus2, cov.control=CovControlMcd())
```

```
pcaogk <- PcaCov(bus2, cov.control=CovControlOgk())

ev    <- getEigenvalues(pca)
evrob <- getEigenvalues(rpca)
evhub <- getEigenvalues(pcaHubert)
evmcd <- getEigenvalues(pcamcd)
evogk <- getEigenvalues(pcaogk)

uvar    <- matrix(nrow=6, ncol=6)
svar    <- sum(ev)
svarrob <- sum(evrob)
svarhub <- sum(evhub)
svarmcd <- sum(evmcd)
svarogk <- sum(evogk)
for(i in 1:6){
    uvar[i,1] <- i
    uvar[i,2] <- round((svar - sum(ev[1:i]))/svar, 3)
    uvar[i,3] <- round((svarrob - sum(evrob[1:i]))/svarrob, 3)
    uvar[i,4] <- round((svarhub - sum(evhub[1:i]))/svarhub, 3)
    uvar[i,5] <- round((svarmcd - sum(evmcd[1:i]))/svarmcd, 3)
    uvar[i,6] <- round((svarogk - sum(evogk[1:i]))/svarogk, 3)
}
uvar <- as.data.frame(uvar)
names(uvar) <- c("q", "Classical","Spherical", "Hubert", "MCD", "OGK")
cat("\nBus data: proportion of unexplained variability for q components\n")
print(uvar)

## Reproduce Table 6.4 from Maronna et al. (2006), page 214
##
## Compute classical and robust PCA extracting only the first 3 components
## and take the squared orthogonal distances to the 3-dimensional hyperplane
##
pca3 <- PcaClassic(bus2, k=3)               # classical
rpca3 <- PcaLocantore(bus2, k=3)            # spherical (Locantore, 1999)
hpca3 <- PcaHubert(bus2, k=3)               # Hubert
dist <- pca3@od^2
rdist <- rpca3@od^2
hdist <- hpca3@od^2

## calculate the quantiles of the distances to the 3-dimensional hyperplane
qclass  <- round(quantile(dist, probs = seq(0, 1, 0.1)[-c(1,11)]), 1)
qspc <- round(quantile(rdist, probs = seq(0, 1, 0.1)[-c(1,11)]), 1)
qhubert <- round(quantile(hdist, probs = seq(0, 1, 0.1)[-c(1,11)]), 1)
qq <- cbind(rbind(qclass, qspc, qhubert), round(c(max(dist), max(rdist), max(hdist)), 0))
colnames(qq)[10] <- "Max"
rownames(qq) <- c("Classical", "Spherical", "Hubert")
cat("\nBus data: quantiles of distances to hiperplane\n")
print(qq)

##
## Reproduce Fig 6.1 from Maronna et al. (2006), page 214
##
cat("\nBus data: Q-Q plot of logs of distances to hyperplane (k=3)
```

```
\nfrom classical and robust estimates. The line is the identity diagonal\n")
plot(sort(log(dist)), sort(log(rdist)), xlab="classical", ylab="robust")
lines(sort(log(dist)), sort(log(dist)))
```

---

bushmiss *Campbell Bushfire Data with added missing data items*

Description

This data set is based on the bushfire data set which was used by Campbell (1984) to locate bushfire scars - see `bushfire` in package `robustbase`. The original dataset contains satelite measurements on five frequency bands, corresponding to each of 38 pixels.

Usage

```
data(bushmiss)
```

Format

A data frame with 190 observations on 6 variables.

The original data set consists of 38 observations in 5 variables. Based on it four new data sets are created in which some of the data items are replaced by missing values with a simple "missing completely at random " mechanism. For this purpose independent Bernoulli trials are realized for each data item with a probability of success 0.1, 0.2, 0.3, 0.4, where success means that the corresponding item is set to missing. The obtained five data sets, including the original one (each with probability of a data item to be missing equal to 0, 0.1, 0.2, 0.3 and 0.4 which is reflected in the new variable `MPROB`) are merged. (See also Beguin and Hulliger (2004).)

Source

Maronna, R.A. and Yohai, V.J. (1995) The Behavoiur of the Stahel-Donoho Robust Multivariate Estimator. *Journal of the American Statistical Association* **90**, 330–341.

Beguin, C. and Hulliger, B. (2004) Multivariate outlier detection in incomplete survey data: the epidemic algorithm and transformed rank correlations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* **127**, 2, 275–294.

Examples

```
## The following code will result in exactly the same output
##  as the one obtained from the original data set
data(bushmiss)
bf <- bushmiss[bushmiss$MPROB==0,1:5]
plot(bf)
covMcd(bf)
```

```
## Not run:
##  This is the code with which the missing data were created:
##
##  Creates a data set with missing values (for testing purposes)
##  from a complete data set 'x'. The probability of
##  each item being missing is 'pr' (Bernoulli trials).
##
getmiss <- function(x, pr=0.1)
{
    n <- nrow(x)
    p <- ncol(x)
    done <- FALSE
    iter <- 0
    while(iter <= 50){
        bt <- rbinom(n*p, 1, pr)
        btmat <- matrix(bt, nrow=n)
        btmiss <- ifelse(btmat==1, NA, 0)
        y <- x+btmiss
        if(length(which(rowSums(nanmap(y)) == p)) == 0)
            return (y)
        iter <- iter + 1
    }
    y
}

## End(Not run)
```

Cascades   *Annual precipitation totals for the North Cascades region*

Description

The data on annual precipitation totals for the North Cascades region contains the sample L-moments ratios (L-CV, L-skewness and L-kurtosis) for 19 sites as used by Hosking and Wallis (1997), page 53, Table 3.4, to illustrate screening tools for regional freqency analysis (RFA).

Usage

```
data(Cascades)
```

Format

A data frame with 19 observations on the following 3 variables.

`L-CV` L-coefficient of variation

`L-skewness` L-coefficient of skewness

`L-kurtosis` L-coefficient of kurtosis

Details

The sample L-moment ratios (L-CV, L-skewness and L-kurtosis) of a site are regarded as a point in three dimensional space.

Source

Hosking, J. R. M. and J. R. Wallis (1997), *Regional Frequency Analysis: An Approach Based on L-moments.* Cambridge University Press, p. 52–53

References

Neykov, N.M., Neytchev, P.N., Van Gelder, P.H.A.J.M. and Todorov V. (2007), Robust detection of discordant sites in regional frequency analysis, *Water Resources Research*, 43, W06417, doi:10.1029/2006WR005322

Examples

```
data(Cascades)

# plot a matrix of scatterplots
pairs(Cascades,
      main="Cascades data set",
      pch=21,
      bg=c("red", "green3", "blue"))

mcd<-CovMcd(Cascades)
mcd
plot(mcd, which="dist", class=TRUE)
plot(mcd, which="dd", class=TRUE)

##  identify the discordant sites using robust distances and compare
##  to the classical ones
rd <- sqrt(getDistance(mcd))
ccov <- CovClassic(Cascades)
cd <- sqrt(getDistance(ccov))
r.out <- which(rd > sqrt(qchisq(0.975,3)))
c.out <- which(cd > sqrt(qchisq(0.975,3)))
cat("Robust: ", length(r.out), " outliers: ", r.out,"\n")
cat("Classical: ", length(c.out), " outliers: ", c.out,"\n")
```

Cov-class — *Class "Cov" – a base class for estimates of multivariate location and scatter*

Description

The class Cov represents an estimate of the multivariate location and scatter of a data set. The objects of class Cov contain the classical estimates and serve as base for deriving other estimates, i.e. different types of robust estimates.

Objects from the Class

Objects can be created by calls of the form `new("Cov",...)`, but the usual way of creating `Cov` objects is a call to the function `Cov` which serves as a constructor.

Slots

`call`**:** Object of class `"language"`

`cov`**:** covariance matrix

`center`**:** location

`n.obs`**:** number of observations used for the computation of the estimates

`mah`**:** mahalanobis distances

`det`**:** determinant

`flag`**:** flags (FALSE if suspected an outlier)

`method`**:** a character string describing the method used to compute the estimate: "Classic"

`singularity`**:** a list with singularity information for the covariance matrix (or `NULL` of not singular)

`X`**:** data

Methods

getCenter `signature(obj = "Cov")`: location vector

getCov `signature(obj = "Cov")`: covariance matrix

getCorr `signature(obj = "Cov")`: correlation matrix

getData `signature(obj = "Cov")`: data frame

getDistance `signature(obj = "Cov")`: distances

getEvals `signature(obj = "Cov")`: Computes and returns the eigenvalues of the covariance matrix

getDet `signature(obj = "Cov")`: Computes and returns the determinant of the covariance matrix (or 0 if the covariance matrix is singular)

getShape `signature(obj = "Cov")`: Computes and returns the shape matrix corresponding to the covariance matrix (i.e. the covariance matrix scaled to have determinant =1)

getFlag `signature(obj = "Cov")`: Flags observations as outliers if the corresponding mahalanobis distance is larger then `qchisq(prob,p)` where `prob` defaults to 0.975.

isClassic `signature(obj = "Cov")`: returns TRUE by default. If necessary, the robust classes will override

plot `signature(x = "Cov")`: plot the object

show `signature(object = "Cov")`: display the object

summary `signature(object = "Cov")`: calculate summary information

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
showClass("Cov")
```

CovClassic *Classical Estimates of Multivariate Location and Scatter*

Description

Computes the classical estimates of multivariate location and scatter. Returns an S4 class `CovClassic` with the estimated `center`, `cov`, Mahalanobis distances and weights based on these distances.

Usage

```
CovClassic(x, unbiased=TRUE)
Cov(x, unbiased=TRUE)
```

Arguments

`x`	a matrix or data frame. As usual, rows are observations and columns are variables.
`unbiased`	whether to return the unbiased estimate of the covariance matrix. Default is `unbiased = TRUE`

Value

An object of class `"CovClassic"`.

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

`Cov-class`, `CovClassic-class`

Examples

```
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
cv <- CovClassic(hbk.x)
cv
summary(cv)
plot(cv)
```

`CovClassic-class` *Class "CovClassic" - classical estimates of multivariate location and scatter*

Description

The class `CovClassic` represents an estimate of the multivariate location and scatter of a data set. The objects of class `CovClassic` contain the classical estimates.

Objects from the Class

Objects can be created by calls of the form `new("CovClassic",...)`, but the usual way of creating `CovClassic` objects is a call to the function `CovClassic` which serves as a constructor.

Slots

`call`**:** Object of class `"language"`

`cov`**:** covariance matrix

`center`**:** location

`n.obs`**:** number of observations used for the computation of the estimates

`mah`**:** mahalanobis distances

`method`**:** a character string describing the method used to compute the estimate: "Classic"

`singularity`**:** a list with singularity information for the ocvariance matrix (or `NULL` of not singular)

`X`**:** data

Methods

getCenter `signature(obj = "CovClassic")`: location vector

getCov `signature(obj = "CovClassic")`: covariance matrix

getCorr `signature(obj = "CovClassic")`: correlation matrix

getData `signature(obj = "CovClassic")`: data frame

getDistance `signature(obj = "CovClassic")`: distances

getEvals `signature(obj = "CovClassic")`: Computes and returns the eigenvalues of the covariance matrix

plot `signature(x = "CovClassic")`: plot the object

show `signature(object = "CovClassic")`: display the object

summary `signature(object = "CovClassic")`: calculate summary information

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
cv <- CovClassic(hbk.x)
cv
summary(cv)
plot(cv)
```

CovControl-class *Class "CovControl" is a VIRTUAL base control class*

Description

The class "CovControl" is a VIRTUAL base control class for the derived classes representing the control parameters for the different robust methods

Arguments

trace	whether to print intermediate results. Default is trace = FALSE
tolSolve	numeric tolerance to be used for inversion (solve) of the covariance matrix in mahalanobis.

Objects from the Class

A virtual Class: No objects may be created from it.

Methods

No methods defined with class "CovControl" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

CovControlMcd *Constructor function for objects of class "CovControlMcd"*

Description

This function will create a control object CovControlMcd containing the control parameters for CovMcd

Usage

```
CovControlMcd(alpha = 0.5, nsamp = 500, scalefn=NULL, maxcsteps=200,
seed = NULL, trace= FALSE, use.correction = TRUE)
```

Arguments

alpha	numeric parameter controlling the size of the subsets over which the determinant is minimized, i.e., alpha*n observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.5.
nsamp	number of subsets used for initial estimates or "best", "exact" or "deterministic". Default is nsamp = 500. For nsamp="best" exhaustive enumeration is done, as long as the number of trials does not exceed 5000. For "exact", exhaustive enumeration will be attempted however many samples are needed. In this case a warning message will be displayed saying that the computation can take a very long time. For "deterministic", the *deterministic* MCD is computed; as proposed by Hubert et al. (2012) it starts from the h most central observations of *six* (deterministic) estimators.
scalefn	function to compute a robust scale estimate or character string specifying a rule determining such a function, see rrcov.control.
maxcsteps	maximal number of concentration steps in the deterministic MCD; should not be reached.
seed	starting value for random generator. Default is seed = NULL
trace	whether to print intermediate results. Default is trace = FALSE
use.correction	whether to use finite sample correction factors. Default is use.correction=TRUE

Value

A CovControlMcd object

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlMcd", alpha=0.75)
ctrl2 <- CovControlMcd(alpha=0.75)

data(hbk)
CovMcd(hbk, control=ctrl1)
```

CovControlMcd-class   *Class 'CovControlMcd' - contains control parameters for CovMcd*

Description

This class extends the CovControl class and contains the control parameters for "CovMcd"

Objects from the Class

Objects can be created by calls of the form new("CovControlMcd",...) or by calling the constructor-function CovControlMcd.

Slots

alpha**:** numeric parameter controlling the size of the subsets over which the determinant is minimized, i.e., alpha*n observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.5.

nsamp number of subsets used for initial estimates or "best", "exact" or "deterministic". Default is nsamp = 500. For nsamp="best" exhaustive enumeration is done, as long as the number of trials does not exceed 5000. For "exact", exhaustive enumeration will be attempted however many samples are needed. In this case a warning message will be displayed saying that the computation can take a very long time.

For "deterministic", the *deterministic* MCD is computed; as proposed by Hubert et al. (2012) it starts from the h most central observations of *six* (deterministic) estimators.

scalefn function to compute a robust scale estimate or character string specifying a rule determining such a function.

maxcsteps maximal number of concentration steps in the deterministic MCD; should not be reached.

seed**:** starting value for random generator. Default is seed = NULL

use.correction**:** whether to use finite sample correction factors. Default is use.correction=TRUE.

trace, tolSolve**:** from the "CovControl" class.

Extends

Class "CovControl", directly.

Methods

restimate `signature(obj = "CovControlMcd")`: the generic function `restimate` allows the different methods for robust estimation to be used polymorphically - this function will call `CovMcd` passing it the control object and will return the obtained `CovRobust` object

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlMcd", alpha=0.75)
ctrl2 <- CovControlMcd(alpha=0.75)

data(hbk)
CovMcd(hbk, control=ctrl1)
```

CovControlMest *Constructor function for objects of class "CovControlMest"*

Description

This function will create a control object `CovControlMest` containing the control parameters for `CovMest`

Usage

```
CovControlMest(r = 0.45, arp = 0.05, eps = 0.001, maxiter = 120)
```

Arguments

`r`	a numeric value specifying the required breakdown point. Allowed values are between `(n - p)/(2 * n)` and 1 and the default is 0.45
`arp`	a numeric value specifying the asympthotic rejection point, i.e. the fraction of points receiving zero weight (see Rocke (1996)). Default is `0.05`
`eps`	a numeric value specifying the relative precision of the solution of the M-estimate. Defaults to `1e-3`
`maxiter`	maximum number of iterations allowed in the computation of the M-estimate. Defaults to 120

Value

A `CovControlMest` object

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlMest", r=0.4)
ctrl2 <- CovControlMest(r=0.4)

data(hbk)
CovMest(hbk, control=ctrl1)
```

`CovControlMest-class` *Class 'CovControlMest' - contains control parameters for "CovMest"*

Description

This class extends the `CovControl` class and contains the control parameters for CovMest

Objects from the Class

Objects can be created by calls of the form `new("CovControlMest",...)` or by calling the constructor-function `CovControlMest`.

Slots

`r`: a numeric value specifying the required breakdown point. Allowed values are between `(n -p)/(2 * n)` and 1 and the default is 0.45

`arp`: a numeric value specifying the asympthotic rejection point, i.e. the fraction of points receiving zero weight (see Rocke (1996)). Default is `0.05`

`eps`: a numeric value specifying the relative precision of the solution of the M-estimate. Defaults to `1e-3`

`maxiter`: maximum number of iterations allowed in the computation of the M-estimate. Defaults to 120

`trace`, `tolSolve`: from the "`CovControl`" class.

Extends

Class "CovControl", directly.

Methods

restimate `signature(obj = "CovControlMest")`: the generic function `restimate` allowes the different methods for robust estimation to be used polymorphically - this function will call `CovMest` passing it the control object and will return the obtained `CovRobust` object

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlMest", r=0.4)
ctrl2 <- CovControlMest(r=0.4)

data(hbk)
CovMest(hbk, control=ctrl1)
```

CovControlMMest — *Constructor function for objects of class "CovControlMMest"*

Description

This function will create a control object `CovControlMMest` containing the control parameters for `CovMMest`

Usage

```
CovControlMMest(bdp = 0.5, eff=0.95, maxiter = 50, sest=CovControlSest(),
    trace = FALSE, tolSolve = 1e-7)
```

Arguments

bdp	a numeric value specifying the required breakdown point. Allowed values are between 0.5 and 1 and the default is 0.5
eff	a numeric value specifying the required efficiency for the MM estimates. Default is `eff=0.95`.

`sest`	an `CovControlSest` object containing control parameters for the initial S-estimate.
`maxiter`	maximum number of iterations allowed in the computation of the MM-estimate. Defaults to 150.
`trace`	whether to print intermediate results. Default is `trace = FALSE`.
`tolSolve`	numeric tolerance to be used as a convergence tolerance for the MM-iteration.

Value

A `CovControlSest` object.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
 ctrl1 <- new("CovControlMMest", bdp=0.25)
 ctrl2 <- CovControlMMest(bdp=0.25)

 data(hbk)
 CovMMest(hbk, control=ctrl1)
```

`CovControlMMest-class` *Class 'CovControlMMest' - contains control parameters for "CovMMest"*

Description

This class extends the `CovControl` class and contains the control parameters for CovMMest

Objects from the Class

Objects can be created by calls of the form `new("CovControlMMest",...)` or by calling the constructor-function `CovControlMMest`.

Slots

bdp a numeric value specifying the required breakdown point. Allowed values are between 0.5 and 1 and the default is `bdp=0.5`.

eff a numeric value specifying the required efficiency for the MM estimates. Default is `eff=0.95`.

sest an `CovControlSest` object containing control parameters for the initial S-estimate.

maxiter maximum number of iterations allowed in the computation of the MM-estimate. Default is `maxiter=50`.

`trace`, `tolSolve`: from the "`CovControl`" class. `tolSolve` is used as a convergence tolerance for the MM-iteration.

Extends

Class "`CovControl`", directly.

Methods

restimate `signature(obj = "CovControlMMest")`: the generic function `restimate` allowes the different methods for robust estimation to be used polymorphically - this function will call `CovMMest` passing it the control object and will return the obtained `CovRobust` object

Author(s)

Valentin Todorov `valentin.todorov@chello.at`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlMMest", bdp=0.25)
ctrl2 <- CovControlMMest(bdp=0.25)

data(hbk)
CovMMest(hbk, control=ctrl1)
```

`CovControlMrcd` *Constructor function for objects of class "CovControlMrcd"*

Description

This function will create a control object `CovControlMrcd` containing the control parameters for `CovMrcd`

Usage

```
CovControlMrcd(alpha = 0.5, h=NULL, maxcsteps=200, rho=NULL,
    target=c("identity", "equicorrelation"), maxcond=50,
    trace= FALSE)
```

Arguments

`alpha`	numeric parameter controlling the size of the subsets over which the determinant is minimized, i.e., `alpha*n` observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.5.
`h`	the size of the subset (can be between ceiling(n/2) and n). Normally NULL and then it `h` will be calculated as `h=ceiling(alpha*n)`. If `h` is provided, `alpha` will be calculated as `alpha=h/n`.
`maxcsteps`	maximal number of concentration steps in the deterministic MCD; should not be reached.
`rho`	regularization parameter. Normally NULL and will be estimated from the data.
`target`	structure of the robust positive definite target matrix: a) "identity": target matrix is diagonal matrix with robustly estimated univariate scales on the diagonal or b) "equicorrelation": non-diagonal target matrix that incorporates an equicorrelation structure (see (17) in paper). Default is `target="identity"`
`maxcond`	maximum condition number allowed (see step 3.4 in algorithm 1). Default is `maxcond=50`
`trace`	whether to print intermediate results. Default is `trace = FALSE`

Value

A `CovControlMrcd` object

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlMrcd", alpha=0.75)
ctrl2 <- CovControlMrcd(alpha=0.75)

data(hbk)
CovMrcd(hbk, control=ctrl1)
```

CovControlMrcd-class *Class 'CovControlMrcd' - contains control parameters for CovMrcd()*

Description

This class extends the `CovControl` class and contains the control parameters for "`CovMrcd`"

Objects from the Class

Objects can be created by calls of the form `new("CovControlMrcd", ...)` or by calling the constructor-function `CovControlMrcd`.

Slots

`alpha`**:** numeric parameter controlling the size of the subsets over which the determinant is minimized, i.e., `alpha*n` observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.5.

h the size of the subset (can be between ceiling(n/2) and n). Normally NULL and then it `h` will be calculated as `h=ceiling(alpha*n)`. If `h` is provided, `alpha` will be calculated as `alpha=h/n`.

maxcsteps maximal number of concentration steps in the deterministic MCD; should not be reached.

rho regularization parameter. Normally NULL and will be estimated from the data.

target structure of the robust positive definite target matrix: a) "identity": target matrix is diagonal matrix with robustly estimated univariate scales on the diagonal or b) "equicorrelation": non-diagonal target matrix that incorporates an equicorrelation structure (see (17) in paper).

maxcond maximum condition number allowed (see step 3.4 in algorithm 1).

`trace`**,** `tolSolve`**:** from the "`CovControl`" class.

Extends

Class "`CovControl`", directly.

Methods

restimate `signature(obj = "CovControlMrcd")`: the generic function `restimate` allows the different methods for robust estimation to be used polymorphically - this function will call `CovMrcd` passing it the control object and will return the obtained `CovRobust` object

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

"CovControlMcd"

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlMrcd", alpha=0.75)
ctrl2 <- CovControlMrcd(alpha=0.75)

data(hbk)
CovMrcd(hbk, control=ctrl1)
```

CovControlMve *Constructor function for objects of class "CovControlMve"*

Description

This function will create a control object CovControlMve containing the control parameters for CovMve

Usage

```
CovControlMve(alpha = 0.5, nsamp = 500, seed = NULL, trace= FALSE)
```

Arguments

alpha	numeric parameter controlling the size of the subsets over which the determinant is minimized, i.e., alpha*n observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.5.
nsamp	number of subsets used for initial estimates or "best" or "exact". Default is nsamp = 500. For nsamp="best" exhaustive enumeration is done, as long as the number of trials does not exceed 5000. For "exact", exhaustive enumeration will be attempted however many samples are needed. In this case a warning message will be displayed saying that the computation can take a very long time.
seed	starting value for random generator. Default is seed = NULL
trace	whether to print intermediate results. Default is trace = FALSE

Value

A CovControlMve object

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlMve", alpha=0.75)
ctrl2 <- CovControlMve(alpha=0.75)

data(hbk)
CovMve(hbk, control=ctrl1)
```

CovControlMve-class *Class 'CovControlMve' - contains control parameters for CovMve*

Description

This class extends the CovControl class and contains the control parameters for "CovMve"

Objects from the Class

Objects can be created by calls of the form new("CovControlMve",...) or by calling the constructor-function CovControlMve.

Slots

alpha: numeric parameter controlling the size of the subsets over which the determinant is minimized, i.e., alpha*n observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.5.

nsamp: number of subsets used for initial estimates or "best" or "exact". Default is nsamp = 500. For nsamp="best" exhaustive enumeration is done, as long as the number of trials does not exceed 5000. For "exact", exhaustive enumeration will be attempted however many samples are needed. In this case a warning message will be displayed saying that the computation can take a very long time.

seed: starting value for random generator. Default is seed = NULL

trace, tolSolve: from the "CovControl" class.

Extends

Class "`CovControl`", directly.

Methods

restimate `signature(obj = "CovControlMve")`: the generic function `restimate` allowes the different methods for robust estimation to be used polymorphically - this function will call `CovMve` passing it the control object and will return the obtained `CovRobust` object

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlMve", alpha=0.75)
ctrl2 <- CovControlMve(alpha=0.75)

data(hbk)
CovMve(hbk, control=ctrl1)
```

`CovControlOgk` *Constructor function for objects of class "CovControlOgk"*

Description

This function will create a control object `CovControlOgk` containing the control parameters for `CovOgk`

Usage

```
CovControlOgk(niter = 2, beta = 0.9, mrob = NULL,
vrob = .vrobGK, smrob = "scaleTau2", svrob = "gk")
```

Arguments

`niter`	number of iterations, usually 1 or 2 since iterations beyond the second do not lead to improvement.
`beta`	coverage parameter for the final reweighted estimate
`mrob`	function for computing the robust univariate location and dispersion - one could use the `tau scale` defined in Yohai and Zamar (1998), see `scaleTau2`. The C version of this function defined by `smrob` is the default.

vrob — function for computing robust estimate of covariance between two random vectors - one could use the function proposed by Gnanadesikan and Kettenring (1972), see covOGK(). The C version of this function defined by svrob is the default.

smrob — a string indicating the name of the function for computing the robust univariate location and dispersion - defaults to scaleTau2 - the scale tau function defined in Yohai and Zamar (1998)

svrob — a string indicating the name of the function for computing robust estimate of covariance between two random vectors - defaults gk, the one proposed by Gnanadesikan and Kettenring (1972)

Details

If the user does not specify a scale and covariance function to be used in the computations or specifies one by using the arguments smrob and svrob (i.e. the names of the functions as strings), a native code written in C will be called which is by far faster than the R version.

If the arguments mrob and vrob are not NULL, the specified functions will be used via the pure R implementation of the algorithm. This could be quite slow.

Value

A CovControlOgk object

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Maronna, R.A. and Zamar, R.H. (2002) Robust estimates of location and dispersion of high-dimensional datasets; *Technometrics* **44**(4), 307–317.

Yohai, R.A. and Zamar, R.H. (1998) High breakdown point estimates of regression by means of the minimization of efficient scale *JASA* **86**, 403–413.

Gnanadesikan, R. and John R. Kettenring (1972) Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics* **28**, 81–124.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlOgk", beta=0.95)
ctrl2 <- CovControlOgk(beta=0.95)

data(hbk)
CovOgk(hbk, control=ctrl1)
```

CovControlOgk-class *Class 'CovControlOgk' - contains control parameters for CovOgk*

Description

This class extends the `CovControl` class and contains the control parameters for `"CovOgk"`

Objects from the Class

Objects can be created by calls of the form `new("CovControlOgk",...)` or by calling the constructor-function `CovControlOgk`.

Slots

niter number of iterations, usually 1 or 2 since iterations beyond the second do not lead to improvement.

beta coverage parameter for the final reweighted estimate

mrob function for computing the robust univariate location and dispersion - defaults to the `tau scale` defined in Yohai and Zamar (1998)

vrob function for computing robust estimate of covariance between two random vectors - defaults the one proposed by Gnanadesikan and Kettenring (1972)

smrob A string indicating the name of the function for computing the robust univariate location and dispersion - defaults to `scaleTau2` - the scale 'tau' function defined in Yohai and Zamar (1998)

svrob A string indicating the name of the function for computing robust estimate of covariance between two random vectors - defaults to gk, the one proposed by Gnanadesikan and Kettenring (1972).

`trace`, `tolSolve`**:** from the `"CovControl"` class.

Extends

Class `"CovControl"`, directly.

Methods

restimate `signature(obj = "CovControlOgk")`: the generic function `restimate` allowes the different methods for robust estimation to be used polymorphically - this function will call `CovOgk` passing it the control object and will return the obtained `CovRobust` object

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlOgk", beta=0.95)
ctrl2 <- CovControlOgk(beta=0.95)

data(hbk)
CovOgk(hbk, control=ctrl1)
```

CovControlSde *Constructor function for objects of class "CovControlSde"*

Description

This function will create a control object CovControlSde containing the control parameters for CovSde

Usage

```
CovControlSde(nsamp = 0, maxres = 0, tune = 0.95, eps = 0.5, prob = 0.99,
    seed = NULL, trace = FALSE, tolSolve = 1e-14)
```

Arguments

nsamp	a positive integer giving the number of resamples required; nsamp may not be reached if too many of the p-subsamples, chosen out of the observed vectors, are in a hyperplane. If nsamp = 0 all possible subsamples are taken. If nsamp is omitted, it is calculated to provide a breakdown point of eps with probability prob.
maxres	a positive integer specifying the maximum number of resamples to be performed including those that are discarded due to linearly dependent subsamples. If maxres is omitted it will be set to 2 times nsamp.
tune	a numeric value between 0 and 1 giving the fraction of the data to receive non-zero weight. Defaults to 0.95.
prob	a numeric value between 0 and 1 specifying the probability of high breakdown point; used to compute nsamp when nsamp is omitted. Defaults to 0.99.
eps	a numeric value between 0 and 0.5 specifying the breakdown point; used to compute nsamp when nresamp is omitted. Defaults to 0.5.
seed	starting value for random generator. Default is seed = NULL.
trace	whether to print intermediate results. Default is trace = FALSE.
tolSolve	numeric tolerance to be used for inversion (solve) of the covariance matrix in mahalanobis.

Value

A CovControlSde object.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlSde", nsamp=2000)
ctrl2 <- CovControlSde(nsamp=2000)

data(hbk)
CovSde(hbk, control=ctrl1)
```

CovControlSde-class   *Class 'CovControlSde' - contains control parameters for "CovSde"*

Description

This class extends the CovControl class and contains the control parameters for CovSde

Objects from the Class

Objects can be created by calls of the form new("CovControlSde",...) or by calling the constructor-function CovControlSde.

Slots

nsamp a positive integer giving the number of resamples required

maxres a positive integer specifying the maximum number of resamples to be performed including those that are discarded due to linearly dependent subsamples.

tune a numeric value between 0 and 1 giving the fraction of the data to receive non-zero weight. Default is tune = 0.95.

prob a numeric value between 0 and 1 specifying the probability of high breakdown point; used to compute nsamp when nsamp is omitted. Default is prob = 0.99.

eps a numeric value between 0 and 0.5 specifying the breakdown point; used to compute nsamp when nresamp is omitted. Default is eps = 0.5.

seed starting value for random generator. Default is seed = NULL.

trace**,** tolSolve**:** from the "CovControl" class.

Extends

Class "CovControl", directly.

Methods

restimate `signature(obj = "CovControlSde")`: ...

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlSde", nsamp=2000)
ctrl2 <- CovControlSde(nsamp=2000)

data(hbk)
CovSde(hbk, control=ctrl1)
```

`CovControlSest` *Constructor function for objects of class "CovControlSest"*

Description

This function will create a control object `CovControlSest` containing the control parameters for `CovSest`

Usage

```
CovControlSest(bdp = 0.5, arp = 0.1, eps = 1e-5, maxiter = 120,
   nsamp = 500, seed = NULL, trace = FALSE, tolSolve = 1e-14, method= "sfast")
```

Arguments

`bdp`	a numeric value specifying the required breakdown point. Allowed values are between `(n -p)/(2 * n)` and 1 and the default is 0.45
`arp`	a numeric value specifying the asympthotic rejection point (for the Rocke type S estimates), i.e. the fraction of points receiving zero weight (see Rocke (1996)). Default is `0.1`
`eps`	a numeric value specifying the relative precision of the solution of the S-estimate (bisquare and Rocke type). Defaults to `1e-5`.
`maxiter`	maximum number of iterations allowed in the computation of the S-estimate (bisquare and Rocke type). Defaults to 120.
`nsamp`	the number of random subsets considered. Default is `nsamp = 500`.

seed	starting value for random generator. Default is seed = NULL.
trace	whether to print intermediate results. Default is trace = FALSE.
tolSolve	numeric tolerance to be used for inversion (solve) of the covariance matrix in mahalanobis.
method	Which algorithm to use: 'sfast'=FAST-S or 'surreal'=SURREAL

Value

A CovControlSest object.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlSest", bdp=0.4)
ctrl2 <- CovControlSest(bdp=0.4)

data(hbk)
CovSest(hbk, control=ctrl1)
```

CovControlSest-class   *Class 'CovControlSest' - contains control parameters for "CovSest"*

Description

This class extends the CovControl class and contains the control parameters for CovSest

Objects from the Class

Objects can be created by calls of the form new("CovControlSest",...) or by calling the constructor-function CovControlSest.

Slots

bdp a numeric value specifying the required breakdown point. Allowed values are between (n -p)/(2 * n) and 1 and the default is bdp=0.45.

arp a numeric value specifying the asympthotic rejection point (for the Rocke type S estimates), i.e. the fraction of points receiving zero weight (see Rocke (1996)). Default is arp=0.1.

eps a numeric value specifying the relative precision of the solution of the S-estimate (bisquare and Rocke type). Default is to `eps=1e-5`.

maxiter maximum number of iterations allowed in the computation of the S-estimate (bisquare and Rocke type). Default is `maxiter=120`.

nsamp the number of random subsets considered. Default is `nsamp = 500`.

seed starting value for random generator. Default is `seed = NULL`.

method Which algorithm to use: 'sfast'=FAST-S, 'surreal'=Ruppert's SURREAL algorithm, 'bisquare'=Bisquare S-estimation with HBDP start or 'rocke' for Rocke type S-estimates

`trace`, `tolSolve`: from the "`CovControl`" class.

Extends

Class "`CovControl`", directly.

Methods

restimate `signature(obj = "CovControlSest")`: the generic function `restimate` allowes the different methods for robust estimation to be used polymorphically - this function will call `CovSest` passing it the control object and will return the obtained `CovRobust` object

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## the following two statements are equivalent
ctrl1 <- new("CovControlSest", bdp=0.4)
ctrl2 <- CovControlSest(bdp=0.4)

data(hbk)
CovSest(hbk, control=ctrl1)
```

`CovMcd` *Robust Location and Scatter Estimation via MCD*

Description

Computes a robust multivariate location and scatter estimate with a high breakdown point, using the 'Fast MCD' (Minimum Covariance Determinant) estimator.

Usage

```
CovMcd(x,
       raw.only=FALSE, alpha=control@alpha, nsamp=control@nsamp,
       scalefn=control@scalefn, maxcsteps=control@maxcsteps,
       initHsets=NULL, save.hsets=FALSE,
       seed=control@seed, trace=control@trace,
       use.correction=control@use.correction,
       control=CovControlMcd(), ...)
```

Arguments

`x`	a matrix or data frame.
`raw.only`	should only the "raw" estimate be returned.
`alpha`	numeric parameter controlling the size of the subsets over which the determinant is minimized, i.e., `alpha*n` observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.5.
`nsamp`	number of subsets used for initial estimates or `"best"`, `"exact"` or `"deterministic"`. Default is `nsamp = 500`. For `nsamp="best"` exhaustive enumeration is done, as long as the number of trials does not exceed 5000. For `"exact"`, exhaustive enumeration will be attempted however many samples are needed. In this case a warning message will be displayed saying that the computation can take a very long time. For `"deterministic"`, the *deterministic* MCD is computed; as proposed by Hubert et al. (2012) it starts from the h most central observations of *six* (deterministic) estimators.
`scalefn`	`function` to compute a robust scale estimate or character string specifying a rule determining such a function, see `rrcov.control`.
`maxcsteps`	maximal number of concentration steps in the deterministic MCD; should not be reached.
`initHsets`	NULL or a Kxh integer matrix of initial subsets of observations of size h (specified by the indices in `1:n`).
`save.hsets`	(for deterministic MCD) logical indicating if the initial subsets should be returned as `initHsets`.
`seed`	starting value for random generator. Default is `seed = NULL`
`trace`	whether to print intermediate results. Default is `trace = FALSE`
`use.correction`	whether to use finite sample correction factors. Default is `use.correction=TRUE`
`control`	a control object (S4) of class `CovControlMcd-class` containing estimation options - same as these provided in the function specification. If the control object is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object.
`...`	potential further arguments passed to **robustbase**'s `covMcd`.

Details

This function computes the minimum covariance determinant estimator of location and scatter and returns an S4 object of class CovMcd-class containing the estimates. The implementation of the function is similar to the existing R function covMcd() which returns an S3 object. The MCD method looks for the $h(> n/2)$ observations (out of n) whose classical covariance matrix has the lowest possible determinant. The raw MCD estimate of location is then the average of these h points, whereas the raw MCD estimate of scatter is their covariance matrix, multiplied by a consistency factor and a finite sample correction factor (to make it consistent at the normal model and unbiased at small samples). Both rescaling factors are returned also in the vector `raw.cnp2` of length 2. Based on these raw MCD estimates, a reweighting step is performed which increases the finite-sample efficiency considerably - see Pison et al. (2002). The rescaling factors for the reweighted estimates are returned in the vector `cnp2` of length 2. Details for the computation of the finite sample correction factors can be found in Pison et al. (2002). The finite sample corrections can be suppressed by setting `use.correction=FALSE`. The implementation in rrcov uses the Fast MCD algorithm of Rousseeuw and Van Driessen (1999) to approximate the minimum covariance determinant estimator.

Value

An S4 object of class CovMcd-class which is a subclass of the virtual class CovRobust-class.

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

P. J. Rousseeuw and A. M. Leroy (1987) *Robust Regression and Outlier Detection.* Wiley.

P. J. Rousseeuw and K. van Driessen (1999) A fast algorithm for the minimum covariance determinant estimator. *Technometrics* **41**, 212–223.

M. Hubert, P. Rousseeuw and T. Verdonck (2012) A deterministic algorithm for robust location and scatter. *Journal of Computational and Graphical Statistics* **21**(3), 618–637.

Pison, G., Van Aelst, S., and Willems, G. (2002), Small Sample Corrections for LTS and MCD, *Metrika*, **55**, 111-123.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

cov.rob from package **MASS**

Examples

```
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
CovMcd(hbk.x)
cD <- CovMcd(hbk.x, nsamp = "deterministic")
summary(cD)
```

```
## the following three statements are equivalent
c1 <- CovMcd(hbk.x, alpha = 0.75)
c2 <- CovMcd(hbk.x, control = CovControlMcd(alpha = 0.75))
## direct specification overrides control one:
c3 <- CovMcd(hbk.x, alpha = 0.75,
             control = CovControlMcd(alpha=0.95))
c1
```

CovMcd-class *MCD Estimates of Multivariate Location and Scatter*

Description

This class, derived from the virtual class "CovRobust" accomodates MCD Estimates of multivariate location and scatter computed by the 'Fast MCD' algorithm.

Objects from the Class

Objects can be created by calls of the form new("CovMcd",...), but the usual way of creating CovMcd objects is a call to the function CovMcd which serves as a constructor.

Slots

alpha: Object of class "numeric" - the size of the subsets over which the determinant is minimized (the default is (n+p+1)/2)

quan: Object of class "numeric" - the number of observations on which the MCD is based. If quan equals n.obs, the MCD is the classical covariance matrix.

best: Object of class "Uvector" - the best subset found and used for computing the raw estimates. The size of best is equal to quan

raw.cov: Object of class "matrix" the raw (not reweighted) estimate of location

raw.center: Object of class "vector" - the raw (not reweighted) estimate of scatter

raw.mah: Object of class "Uvector" - mahalanobis distances of the observations based on the raw estimate of the location and scatter

raw.wt: Object of class "Uvector" - weights of the observations based on the raw estimate of the location and scatter

raw.cnp2: Object of class "numeric" - a vector of length two containing the consistency correction factor and the finite sample correction factor of the raw estimate of the covariance matrix

cnp2: Object of class "numeric" - a vector of length two containing the consistency correction factor and the finite sample correction factor of the final estimate of the covariance matrix.

iter, crit, wt: from the "CovRobust" class.

call, cov, center, n.obs, mah, method, singularity, X: from the "Cov" class.

Extends

Class "CovRobust", directly. Class "Cov", by class "CovRobust".

Methods

No methods defined with class "CovMcd" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

CovMcd, Cov-class, CovRobust-class

Examples

```
showClass("CovMcd")
```

CovMest *Constrained M-Estimates of Location and Scatter*

Description

Computes constrained M-Estimates of multivariate location and scatter based on the translated biweight function ('t-biweight') using a High breakdown point initial estimate as defined by Rocke (1996). The default initial estimate is the Minimum Volume Ellipsoid computed with CovMve. The raw (not reweighted) estimates are taken and the covariance matrix is standardized to determinant 1.

Usage

```
CovMest(x, r = 0.45, arp = 0.05, eps=1e-3,
    maxiter=120, control, t0, S0, initcontrol)
```

Arguments

x	a matrix or data frame.
r	required breakdown point. Allowed values are between (n -p)/(2 * n) and 1 and the default is 0.45
arp	asympthotic rejection point, i.e. the fraction of points receiving zero weight (see Rocke (1996)). Default is 0.05.
eps	a numeric value specifying the relative precision of the solution of the M-estimate. Defaults to 1e-3
maxiter	maximum number of iterations allowed in the computation of the M-estimate. Defaults to 120

control	a control object (S4) of class CovControlMest-class containing estimation options - same as these provided in the fucntion specification. If the control object is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object.
t0	optional initial high breakdown point estimates of the location. If not supplied MVE will be used.
S0	optional initial high breakdown point estimates of the scatter. If not supplied MVE will be used.
initcontrol	optional control object - of class CovControl - specifing the initial high breakdown point estimates of location and scatter. If not supplied MVE will be used.

Details

Rocke (1996) has shown that the S-estimates of multivariate location and scatter in high dimensions can be sensitive to outliers even if the breakdown point is set to be near 0.5. To mitigate this problem he proposed to utilize the translated biweight (or t-biweight) method with a standardization step consisting of equating the median of `rho(d)` with the median under normality. This is then not an S-estimate, but is instead a constrained M-estimate. In order to make the smooth estimators to work, a reasonable starting point is necessary, which will lead reliably to a good solution of the estimator. In `CovMest` the MVE computed by `CovMve` is used, but the user has the possibility to give her own initial estimates.

Value

An object of class `CovMest-class` which is a subclass of the virtual class `CovRobust-class`.

Note

The psi, rho and weight functions for the M estimation are encapsulated in a virtual S4 class `PsiFun` from which a `PsiBwt` class, implementing the translated biweight (t-biweight), is dervied. The base class `PsiFun` contains also the M-iteration itself. Although not documented and not accessibale directly by the user these classes will form the bases for adding other functions (biweight, LWS, etc.) as well as S-estimates.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>,

(some code from C. Becker - http://www.sfb475.uni-dortmund.de/dienst/de/content/struk-d/bereicha-d/tpa1softw-d.html)

References

D.L.Woodruff and D.M.Rocke (1994) Computable robust estimation of multivariate location and shape on high dimension using compound estimators, *Journal of the American Statistical Association*, **89**, 888–896.

D.M.Rocke (1996) Robustness properties of S-estimates of multivariate location and shape in high dimension, *Annals of Statistics*, **24**, 1327-1345.

D.M.Rocke and D.L.Woodruff (1996) Identification of outliers in multivariate data *Journal of the American Statistical Association*, **91**, 1047–1061.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

covMcd, Cov-class, CovMve, CovRobust-class, CovMest-class

Examples

```
library(rrcov)
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
CovMest(hbk.x)

## the following four statements are equivalent
c0 <- CovMest(hbk.x)
c1 <- CovMest(hbk.x, r = 0.45)
c2 <- CovMest(hbk.x, control = CovControlMest(r = 0.45))
c3 <- CovMest(hbk.x, control = new("CovControlMest", r = 0.45))

## direct specification overrides control one:
c4 <- CovMest(hbk.x, r = 0.40,
             control = CovControlMest(r = 0.25))
c1
summary(c1)
plot(c1)
```

---

covMest *Constrained M-Estimates of Location and Scatter*

Description

Computes constrained M-Estimates of multivariate location and scatter based on the translated biweight function (‘t-biweight’) using a High breakdown point initial estimate. The default initial estimate is the Minimum Volume Ellipsoid computed with CovMve. The raw (not reweighted) estimates are taken and the covariance matrix is standardized to determinant 1.

Usage

```
covMest(x, cor=FALSE, r = 0.45, arp = 0.05, eps=1e-3,
    maxiter=120, control, t0, S0)
```

Arguments

`x`	a matrix or data frame.
`cor`	should the returned result include a correlation matrix? Default is `cor = FALSE`.
`r`	required breakdown point. Allowed values are between `(n - p)/(2 * n)` and 1 and the default is 0.45
`arp`	asympthotic rejection point, i.e. the fraction of points receiving zero weight (see Rocke (1996)). Default is `0.05`.
`eps`	a numeric value specifying the relative precision of the solution of the M-estimate. Defaults to `1e-3`
`maxiter`	maximum number of iterations allowed in the computation of the M-estimate. Defaults to 120
`control`	a list with estimation options - same as these provided in the fucntion specification. If the control object is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object.
`t0`	optional initial high breakdown point estimates of the location. If not supplied MVE will be used.
`S0`	optional initial high breakdown point estimates of the scatter. If not supplied MVE will be used.

Details

Rocke (1996) has shown that the S-estimates of multivariate location and scatter in high dimensions can be sensitive to outliers even if the breakdown point is set to be near 0.5. To mitigate this problem he proposed to utilize the translated biweight (or t-biweight) method with a standardization step consisting of equating the median of `rho(d)` with the median under normality. This is then not an S-estimate, but is instead a constrained M-estimate. In order to make the smooth estimators to work, a reasonable starting point is necessary, which will lead reliably to a good solution of the estimator. In `covMest` the MVE computed by `CovMve` is used, but the user has the possibility to give her own initial estimates.

Value

An object of class `"mest"` which is basically a `list` with the following components. This class is "derived" from `"mcd"` so that the same generic functions - `print`, `plot`, `summary` - can be used. NOTE: this is going to change - in one of the next revisions `covMest` will return an S4 class `"mest"` which is derived (i.e. `contains`) form class `"cov"`.

`center`	the final estimate of location.
`cov`	the final estimate of scatter.
`cor`	the estimate of the correlation matrix (only if `cor = TRUE`).
`mah`	mahalanobis distances of the observations using the M-estimate of the location and scatter.
`X`	the input data as a matrix.
`n.obs`	total number of observations.

`method`	character string naming the method (M-Estimates).
`call`	the call used (see `match.call`).

Note

The psi, rho and weight functions for the M estimation are encapsulated in a virtual S4 class `PsiFun` from which a `PsiBwt` class, implementing the translated biweight (t-biweight), is dervied. The base class `PsiFun` contains also the M-iteration itself. Although not documented and not accessibale directly by the user these classes will form the bases for adding other functions (biweight, LWS, etc.) as well as S-estimates.

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`,

(some code from C. Becker - http://www.sfb475.uni-dortmund.de/dienst/de/content/struk-d/bereicha-d/tpa1softw-d.html)

References

D.L.Woodruff and D.M.Rocke (1994) Computable robust estimation of multivariate location and shape on high dimension using compound estimators, *Journal of the American Statistical Association*, **89**, 888–896.

D.M.Rocke (1996) Robustness properties of S-estimates of multivariate location and shape in high dimension, *Annals of Statistics*, **24**, 1327-1345.

D.M.Rocke and D.L.Woodruff (1996) Identification of outliers in multivariate data *Journal of the American Statistical Association*, **91**, 1047–1061.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

See Also

`covMcd`

`CovMest-class` *Constrained M-estimates of Multivariate Location and Scatter*

Description

This class, derived from the virtual class "CovRobust" accomodates constrained M-Estimates of multivariate location and scatter based on the translated biweight function (‘t-biweight’) using a High breakdown point initial estimate (Minimum Covariance Determinant - ‘Fast MCD’)

Objects from the Class

Objects can be created by calls of the form `new("CovMest",...)`, but the usual way of creating `CovMest` objects is a call to the function `CovMest` which serves as a constructor.

Slots

`vt`: Object of class "vector" - vector of weights (v)

`iter`, `crit`, `wt`: from the "CovRobust" class.

`call`, `cov`, `center`, `n.obs`, `mah`, `method`, `singularity`, `X`: from the "Cov" class.

Extends

Class "CovRobust", directly. Class "Cov", by class "CovRobust".

Methods

No methods defined with class "CovMest" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

CovMest, Cov-class, CovRobust-class

Examples

```
showClass("CovMest")
```

CovMMest *MM Estimates of Multivariate Location and Scatter*

Description

Computes MM-Estimates of multivariate location and scatter starting from an initial S-estimate

Usage

```
CovMMest(x, bdp = 0.5, eff = 0.95, eff.shape=TRUE, maxiter = 50,
    trace = FALSE, tolSolve = 1e-7, control)
```

Arguments

`x`	a matrix or data frame.
`bdp`	a numeric value specifying the required breakdown point. Allowed values are between 0.5 and 1 and the default is `bdp=0.5`.
`eff`	a numeric value specifying the required efficiency for the MM estimates. Default is `eff=0.95`.
`eff.shape`	logical; if TRUE, eff is with regard to shape-efficiency, otherwise location-efficiency. Default is `eff.shape=FALSE`.
`maxiter`	maximum number of iterations allowed in the computation of the S-estimate (bisquare and Rocke type). Default is `maxiter=50`.
`trace`	whether to print intermediate results. Default is `trace = FALSE`.
`tolSolve`	numeric tolerance to be used as a convergence tolerance for the MM-iteration
`control`	a control object (S4) of class `CovControlMMest-class` containing estimation options - same as these provided in the fucntion specification. If the control object is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object.

Details

Computes MM-estimates of multivariate location and scatter starting from an initial S-estimate.

Value

An S4 object of class `CovMMest-class` which is a subclass of the virtual class `CovRobust-class`.

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Tatsuoka, K.S. and Tyler, D.E. (2000). The uniqueness of S and M-functionals under non-elliptical distributions. *Annals of Statistics* 28, 1219–1243

M. Salibian-Barrera, S. Van Aelstt and G. Willems (2006). Principal components analysis based on multivariate MM-estimators with fast and robust bootstrap. *Journal of the American Statistical Association* 101, 1198–1211.

R. A. Maronna, D. Martin and V. Yohai (2006). *Robust Statistics: Theory and Methods*. Wiley, New York.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

Examples

```
library(rrcov)
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
CovMMest(hbk.x)

## the following four statements are equivalent
c0 <- CovMMest(hbk.x)
c1 <- CovMMest(hbk.x, bdp = 0.25)
c2 <- CovMMest(hbk.x, control = CovControlMMest(bdp = 0.25))
c3 <- CovMMest(hbk.x, control = new("CovControlMMest", bdp = 0.25))

## direct specification overrides control one:
c4 <- CovMMest(hbk.x, bdp = 0.40,
             control = CovControlMMest(bdp = 0.25))
c1
summary(c1)
plot(c1)

## Deterministic MM-estmates
CovMMest(hbk.x, control=CovControlMMest(sest=CovControlSest(method="sdet")))
```

CovMMest-class *MM Estimates of Multivariate Location and Scatter*

Description

This class, derived from the virtual class "CovRobust" accomodates MM Estimates of multivariate location and scatter.

Objects from the Class

Objects can be created by calls of the form new("CovMMest",...), but the usual way of creating CovSest objects is a call to the function CovMMest which serves as a constructor.

Slots

det, flag, iter, crit: from the "CovRobust" class.

c1 tuning parameter of the loss function for MM-estimation (depend on control parameters eff and eff.shape). Can be computed by the internal function .csolve.bw.MM(p,eff,eff.shape=TRUE). For the tuning parameters of the underlying S-estimate see the slot sest and "CovSest".

sest an CovSest object containing the initial S-estimate.

call, cov, center, n.obs, mah, method, singularity, X: from the "Cov" class.

Extends

Class "CovRobust", directly. Class "Cov", by class "CovRobust".

Methods

No methods defined with class "CovMMest" in the signature.

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

See Also

`CovMMest`, `Cov-class`, `CovRobust-class`

Examples

```
showClass("CovMMest")
```

CovMrcd — *Robust Location and Scatter Estimation via Minimum Regularized Covariance Determonant (MRCD)*

Description

Computes a robust multivariate location and scatter estimate with a high breakdown point, using the Minimum Regularized Covariance Determonant (MRCD) estimator.

Usage

```
CovMrcd(x,
        alpha=control@alpha,
        h=control@h,
        maxcsteps=control@maxcsteps,
        initHsets=NULL, save.hsets=FALSE,
        rho=control@rho,
        target=control@target,
        maxcond=control@maxcond,
        trace=control@trace,
        control=CovControlMrcd())
```

Arguments

`x` a matrix or data frame.

`alpha` numeric parameter controlling the size of the subsets over which the determinant is minimized, i.e., `alpha*n` observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.5.

`h` the size of the subset (can be between ceiling(n/2) and n). Normally NULL and then it `h` will be calculated as `h=ceiling(alpha*n)`. If `h` is provided, `alpha` will be calculated as `alpha=h/n`.

`maxcsteps` maximal number of concentration steps in the deterministic MCD; should not be reached.

`initHsets` NULL or a Kxh integer matrix of initial subsets of observations of size h (specified by the indices in `1:n`).

`save.hsets` (for deterministic MCD) logical indicating if the initial subsets should be returned as `initHsets`.

`rho` regularization parameter. Normally NULL and will be estimated from the data.

`target` structure of the robust positive definite target matrix: a) "identity": target matrix is diagonal matrix with robustly estimated univariate scales on the diagonal or b) "equicorrelation": non-diagonal target matrix that incorporates an equicorrelation structure (see (17) in paper). Default is `target="identity"`

`maxcond` maximum condition number allowed (see step 3.4 in algorithm 1). Default is `maxcond=50`

`trace` whether to print intermediate results. Default is `trace = FALSE`

`control` a control object (S4) of class `CovControlMrcd-class` containing estimation options - same as these provided in the function specification. If the control object is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object.

Details

This function computes the minimum regularized covariance determinant estimator (MRCD) of location and scatter and returns an S4 object of class `CovMrcd-class` containing the estimates. Similarly like the MCD method, MRCD looks for the $h(> n/2)$ observations (out of n) whose classical covariance matrix has the lowest possible determinant, but replaces the subset-based covariance by a regularized covariance estimate, defined as a weighted average of the sample covariance of the h-subset and a predetermined positive definite target matrix. The Minimum Regularized Covariance Determinant (MRCD) estimator is then the regularized covariance based on the h-subset which makes the overall determinant the smallest. A data-driven procedure sets the weight of the target matrix (`rho`), so that the regularization is only used when needed.

Value

An S4 object of class `CovMrcd-class` which is a subclass of the virtual class `CovRobust-class`.

Author(s)

Kris Boudt, Peter Rousseeuw, Steven Vanduffel and Tim Verdonk. Improved by Joachim Schreurs and Iwein Vranckx. Adapted for **rrcov** by Valentin Todorov <valentin.todorov@chello.at>

References

Kris Boudt, Peter Rousseeuw, Steven Vanduffel and Tim Verdonck (2018) The Minimum Regularized Covariance Determinant estimator. submitted, available at https://arxiv.org/abs/1701.07086.

Mia Hubert, Peter Rousseeuw and Tim Verdonck (2012) A deterministic algorithm for robust location and scatter. *Journal of Computational and Graphical Statistics* **21**(3), 618–637.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

CovMcd

Examples

```
## The result will be (almost) identical to the raw MCD
##  (since we do not do reweighting of MRCD)
##
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
c0 <- CovMcd(hbk.x, alpha=0.75, use.correction=FALSE)
cc <- CovMrcd(hbk.x, alpha=0.75)
cc$rho
all.equal(c0$best, cc$best)
all.equal(c0$raw.center, cc$center)
all.equal(c0$raw.cov/c0$raw.cnp2[1], cc$cov/cc$cnp2)

summary(cc)

## the following three statements are equivalent
c1 <- CovMrcd(hbk.x, alpha = 0.75)
c2 <- CovMrcd(hbk.x, control = CovControlMrcd(alpha = 0.75))
## direct specification overrides control one:
c3 <- CovMrcd(hbk.x, alpha = 0.75,
             control = CovControlMrcd(alpha=0.95))
c1

## Not run:
data(octane)
n <- nrow(octane)
p <- ncol(octane)
out <- CovMrcd(octane, h=33)
robpca = PcaHubert(octane, k=2, alpha=0.75, mcd=FALSE)
(outl.robpca = which(robpca@flag==FALSE))
```

```
# Observations flagged as outliers by ROBPCA:
# 25, 26, 36, 37, 38, 39

# Plot the orthogonal distances versus the score distances:
pch = rep(20,n); pch[robpca@flag==FALSE] = 17
col = rep('black',n); col[robpca@flag==FALSE] = 'red'
plot(robpca, pch=pch, col=col, id.n.sd=6, id.n.od=6)

## Plot now the MRCD mahalanobis distances
pch = rep(20,n); pch[!getFlag(out)] = 17
col = rep('black',n); col[!getFlag(out)] = 'red'
plot(out, pch=pch, col=col, id.n=6)

## End(Not run)
```

CovMrcd-class *MRCD Estimates of Multivariate Location and Scatter*

Description

This class, derived from the virtual class "CovRobust" accomodates MRCD Estimates of multivariate location and scatter computed by a variant of the 'Fast MCD' algorithm.

Objects from the Class

Objects can be created by calls of the form new("CovMrcd",...), but the usual way of creating CovMrcd objects is a call to the function CovMrcd which serves as a constructor.

Slots

alpha: Object of class "numeric" - the size of the subsets over which the determinant is minimized (the default is (n+p+1)/2)

quan: Object of class "numeric" - the number of observations on which the MCD is based. If quan equals n.obs, the MCD is the classical covariance matrix.

best: Object of class "Uvector" - the best subset found and used for computing the raw estimates. The size of best is equal to quan

cnp2: Object of class "numeric" - containing the consistency correction factor of the estimate of the covariance matrix.

icov: The inverse of the covariance matrix.

rho: The estimated regularization parameter.

target: The estimated target matrix.

crit: from the "CovRobust" class.

call, cov, center, n.obs, mah, method, X: from the "Cov" class.

Extends

Class "CovRobust", directly. Class "Cov", by class "CovRobust".

Methods

No methods defined with class "CovMrcd" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

CovMrcd, Cov-class, CovRobust-class, CovMcd-class

Examples

```
showClass("CovMrcd")
```

CovMve *Robust Location and Scatter Estimation via MVE*

Description

Computes a robust multivariate location and scatter estimate with a high breakdown point, using the 'MVE' (Minimum Volume Ellipsoid) estimator.

Usage

```
CovMve(x, alpha = 1/2, nsamp = 500, seed = NULL, trace = FALSE, control)
```

Arguments

x	a matrix or data frame.
alpha	numeric parameter controlling the size of the subsets over which the determinant is minimized, i.e., alpha*n observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.5.
nsamp	number of subsets used for initial estimates or "best" or "exact". Default is nsamp = 500. For nsamp="best" exhaustive enumeration is done, as long as the number of trials does not exceed 5000. For "exact", exhaustive enumeration will be attempted however many samples are needed. In this case a warning message will be displayed saying that the computation can take a very long time.
seed	starting value for random generator. Default is seed = NULL
trace	whether to print intermediate results. Default is trace = FALSE

`control` a control object (S4) of class `CovControlMve-class` containing estimation options - same as these provided in the fucntion specification. If the control object is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object.

Details

This function computes the minimum volume ellipsoid estimator of location and scatter and returns an S4 object of class `CovMve-class` containing the estimates.

The approximate estimate is based on a subset of size `alpha*n` with an enclosing ellipsoid of smallest volume. The mean of the best found subset provides the raw estimate of the location, and the rescaled covariance matrix is the raw estimate of scatter. The rescaling of the raw covariance matrix is by `median(dist)/qchisq(0.5,p)` and this scale factor is returned in the slot `raw.cnp2`. Currently no finite sample corrction factor is applied. The Mahalanobis distances of all observations from the location estimate for the raw covariance matrix are calculated, and those points within the 97.5 under Gaussian assumptions are declared to be good. The final (reweightd) estimates are the mean and rescaled covariance of the good points. The reweighted covariance matrix is rescaled by `1/pgamma(qchisq(alpha,p)/2,p/2 + 1)/alpha` (see Croux and Haesbroeck, 1999) and this scale factor is returned in the slot `cnp2`.

The search for the approximate solution is made over ellipsoids determined by the covariance matrix of `p+1` of the data points and applying a simple but effective improvement of the subsampling procedure as described in Maronna et al. (2006), p. 198. Although there exists no formal proof of this improvement (as for MCD and LTS), simulations show that it can be recommended as an approximation of the MVE.

Value

An S4 object of class `CovMve-class` which is a subclass of the virtual class `CovRobust-class`.

Note

Main reason for implementing the MVE estimate was that it is the recommended initial estimate for S estimation (see Maronna et al. (2006), p. 199) and will be used by default in `CovMest` (after removing the correction factors from the covariance matrix and rescaling to determinant 1).

Author(s)

Valentin Todorov <valentin.todorov@chello.at> and Matias Salibian-Barrera <matias@stat.ubc.ca>

References

P. J. Rousseeuw and A. M. Leroy (1987) *Robust Regression and Outlier Detection.* Wiley.

C. Croux and G. Haesbroeck (1999). Influence function and efficiency of the minimum covariance determinant scatter matrix estimator. *Journal of Multivariate Analysis*, **71**, 161–190.

R. A. Maronna, D. Martin and V. Yohai (2006). *Robust Statistics: Theory and Methods*. Wiley, New York.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

cov.rob from package **MASS**

Examples

```
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
CovMve(hbk.x)

## the following three statements are equivalent
c1 <- CovMve(hbk.x, alpha = 0.75)
c2 <- CovMve(hbk.x, control = CovControlMve(alpha = 0.75))
## direct specification overrides control one:
c3 <- CovMve(hbk.x, alpha = 0.75,
             control = CovControlMve(alpha=0.95))
c1
```

CovMve-class *MVE Estimates of Multivariate Location and Scatter*

Description

This class, derived from the virtual class "CovRobust" accomodates MVE Estimates of multivariate location and scatter computed by the 'Fast MVE' algorithm.

Objects from the Class

Objects can be created by calls of the form new("CovMve",...), but the usual way of creating CovMve objects is a call to the function CovMve which serves as a constructor.

Slots

alpha**:** Object of class "numeric" - the size of the subsets over which the volume of the ellipsoid is minimized (the default is (n+p+1)/2)

quan**:** Object of class "numeric" - the number of observations on which the MVE is based. If quan equals n.obs, the MVE is the classical covariance matrix.

best**:** Object of class "Uvector" - the best subset found and used for computing the raw estimates. The size of best is equal to quan

raw.cov**:** Object of class "matrix" the raw (not reweighted) estimate of location

raw.center**:** Object of class "vector" - the raw (not reweighted) estimate of scatter

raw.mah**:** Object of class "Uvector" - mahalanobis distances of the observations based on the raw estimate of the location and scatter

raw.wt**:** Object of class "Uvector" - weights of the observations based on the raw estimate of the location and scatter

raw.cnp2**:** Object of class "numeric" - a vector of length two containing the consistency correction factor and the finite sample correction factor of the raw estimate of the covariance matrix cnp2: Object of class "numeric" - a vector of length two containing the consistency correction factor and the finite sample correction factor of the final estimate of the covariance matrix.

iter, crit, wt: from the "CovRobust" class.

call, cov, center, n.obs, mah, method, singularity, X: from the "Cov" class.

Extends

Class "CovRobust", directly. Class "Cov", by class "CovRobust".

Methods

No methods defined with class "CovMve" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

CovMve, Cov-class, CovRobust-class

Examples

```
showClass("CovMve")
```

CovOgk *Robust Location and Scatter Estimation - Ortogonalized Gnanadesikan-Kettenring (OGK)*

Description

Computes a robust multivariate location and scatter estimate with a high breakdown point, using the pairwise algorithm proposed by Marona and Zamar (2002) which in turn is based on the pairwise robust estimator proposed by Gnanadesikan-Kettenring (1972).

Usage

```
CovOgk(x, niter = 2, beta = 0.9, control)
```

Arguments

- `x` — a matrix or data frame.
- `niter` — number of iterations, usually 1 or 2 since iterations beyond the second do not lead to improvement.
- `beta` — coverage parameter for the final reweighted estimate
- `control` — a control object (S4) of class `CovControlOgk-class` containing estimation options - same as these provided in the function specification. If the control object is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object. The control object contains also functions for computing the robust univariate location and dispersion estimate `mrob` and for computing the robust estimate of the covariance between two random variables `vrob`.

Details

The method proposed by Marona and Zamar (2002) allowes to obtain positive-definite and almost affine equivariant robust scatter matrices starting from any pairwise robust scatter matrix. The default robust estimate of covariance between two random vectors used is the one proposed by Gnanadesikan and Kettenring (1972) but the user can choose any other method by redefining the function in slot `vrob` of the control object `CovControlOgk`. Similarly, the function for computing the robust univariate location and dispersion used is the `tau scale` defined in Yohai and Zamar (1998) but it can be redefined in the control object.

The estimates obtained by the OGK method, similarly as in `CovMcd` are returned as 'raw' estimates. To improve the estimates a reweighting step is performed using the coverage parameter `beta` and these reweighted estimates are returned as 'final' estimates.

Value

An S4 object of class `CovOgk-class` which is a subclass of the virtual class `CovRobust-class`.

Note

If the user does not specify a scale and covariance function to be used in the computations or specifies one by using the arguments `smrob` and `svrob` (i.e. the names of the functions as strings), a native code written in C will be called which is by far faster than the R version.

If the arguments `mrob` and `vrob` are not NULL, the specified functions will be used via the pure R implementation of the algorithm. This could be quite slow.

See `CovControlOgk` for details.

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>` and Kjell Konis `<kjell.konis@epfl.ch>`

References

Maronna, R.A. and Zamar, R.H. (2002) Robust estimates of location and dispersion of high-dimensional datasets; *Technometrics* **44**(4), 307–317.

Yohai, R.A. and Zamar, R.H. (1998) High breakdown point estimates of regression by means of the minimization of efficient scale *JASA* **86**, 403–413.

Gnanadesikan, R. and John R. Kettenring (1972) Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics* **28**, 81–124.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

CovMcd, CovMest

Examples

```
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
CovOgk(hbk.x)

## the following three statements are equivalent
c1 <- CovOgk(hbk.x, niter=1)
c2 <- CovOgk(hbk.x, control = CovControlOgk(niter=1))

## direct specification overrides control one:
c3 <- CovOgk(hbk.x, beta=0.95,
             control = CovControlOgk(beta=0.99))
c1

x<-matrix(c(1,2,3,7,1,2,3,7), ncol=2)
##  CovOgk(x)   - this would fail because the two columns of x are exactly collinear.
##              In order to fix it, redefine the default 'vrob' function for example
##              in the following way and pass it as a parameter in the control
##              object.
cc <- CovOgk(x, control=new("CovControlOgk",
                            vrob=function(x1, x2, ...)
                            {
                                r <- .vrobGK(x1, x2, ...)
                                if(is.na(r))
                                    r <- 0
                                r
                            })
)
cc
```

CovOgk-class — *OGK Estimates of Multivariate Location and Scatter*

Description

This class, derived from the virtual class "CovRobust" accomodates OGK Estimates of multivariate location and scatter computed by the algorithm proposed by Marona and Zamar (2002).

Objects from the Class

Objects can be created by calls of the form new("CovOgk",...), but the usual way of creating CovOgk objects is a call to the function CovOgk which serves as a constructor.

Slots

raw.cov: Object of class "matrix" the raw (not reweighted) estimate of covariance matrix

raw.center: Object of class "vector" - the raw (not reweighted) estimate of the location vector

raw.mah: Object of class "Uvector" - mahalanobis distances of the observations based on the raw estimate of the location and scatter

raw.wt: Object of class "Uvector" - weights of the observations based on the raw estimate of the location and scatter

iter, crit, wt: from the "CovRobust" class.

call, cov, center, n.obs, mah, method, singularity, X: from the "Cov" class.

Extends

Class "CovRobust", directly. Class "Cov", by class "CovRobust".

Methods

No methods defined with class "CovOgk" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

CovMcd-class, CovMest-class

Examples

```
showClass("CovOgk")
```

CovRobust *Robust Location and Scatter Estimation*

Description

Computes a robust multivariate location and scatter estimate with a high breakdown point, using one of the available estimators.

Usage

```
CovRobust(x, control, na.action = na.fail)
```

Arguments

`x`	a matrix or data frame.
`control`	a control object (S4) for one of the available control classes, e.g. `CovControlMcd-class`, `CovControlOgk-class`, `CovControlSest-class`, etc., containing estimation options. The class of this object defines which estimator will be used. Alternatively a character string can be specified which names the estimator - one of auto, sde, mcd, ogk, m, mve, sfast, surreal, bisquare, rocke. If 'auto' is specified or the argument is missing, the function will select the estimator (see below for details)
`na.action`	A function to specify the action to be taken if missing values are found. The default action is for the procedure to fail. An alternative is na.omit, which leads to rejection of cases with missing values on any required variable.

Details

This function simply calls the `restimate` method of the control object `control`. If a character string naming an estimator is specified, a new control object will be created and used (with default estimation options). If this argument is missing or a character string 'auto' is specified, the function will select the robust estimator according to the size of the dataset. If there are less than 1000 observations and less than 10 variables or less than 5000 observations and less than 5 variables, Stahel-Donoho estimator will be used. Otherwise, if there are less than 50000 observations either bisquare S-estimates (for less than 10 variables) or Rocke type S-estimates (for 10 to 20 variables) will be used. In both cases the S iteration starts at the initial MVE estimate. And finally, if there are more than 50000 observations and/or more than 20 variables the Orthogonalized Quadrant Correlation estimator (`CovOgk` with the corresponding parameters) is used.

Value

An object derived from a `CovRobust` object, depending on the selected estimator.

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
CovRobust(hbk.x)
CovRobust(hbk.x, CovControlSest(method="bisquare"))
```

CovRobust-class — *Class "CovRobust" - virtual base class for robust estimates of multivariate location and scatter*

Description

CovRobust is a virtual base class used for deriving the concrete classes representing different robust estimates of multivariate location and scatter. Here are implemeted the standard methods common for all robust estimates like show, summary and plot. The derived classes can override these methods and can define new ones.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

iter: number of iterations used to compute the estimates

crit: value of the criterion function

wt: weights

call, cov, center, n.obs, mah, method, singularity, X: from the "Cov" class.

Extends

Class "Cov", directly.

Methods

isClassic signature(obj = "CovRobust"): Will return FALSE, since this is a 'Robust' object

getMeth signature(obj = "CovRobust"): Return the name of the particular robust method used (as a character string)

show signature(object = "CovRobust"): display the object

plot signature(x = "CovRobust"): plot the object

getRaw signature(obj = "CovRobust"): Return the object with the reweighted estimates replaced by the raw ones (only relevant for CovMcd, CovMve and CovOgk)

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

Cov-class, CovMcd-class, CovMest-class, CovOgk-class

Examples

```
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
cv <- CovMest(hbk.x)                # it is not possible to create an object of
                                    # class CovRobust, since it is a VIRTUAL class
cv
summary(cv)                         # summary method for class CovRobust
plot(cv)                            # plot method for class CovRobust
```

CovSde *Stahel-Donoho Estimates of Multivariate Location and Scatter*

Description

Compute a robust estimate of location and scale using the Stahel-Donoho projection based estimator

Usage

```
CovSde(x, nsamp, maxres, tune = 0.95, eps = 0.5, prob = 0.99,
seed = NULL, trace = FALSE, control)
```

Arguments

x	a matrix or data frame.
nsamp	a positive integer giving the number of resamples required; nsamp may not be reached if too many of the p-subsamples, chosen out of the observed vectors, are in a hyperplane. If nsamp = 0 all possible subsamples are taken. If nsamp is omitted, it is calculated to provide a breakdown point of eps with probability prob.
maxres	a positive integer specifying the maximum number of resamples to be performed including those that are discarded due to linearly dependent subsamples. If maxres is omitted it will be set to 2 times nsamp.
tune	a numeric value between 0 and 1 giving the fraction of the data to receive non-zero weight. Defaults to 0.95

prob	a numeric value between 0 and 1 specifying the probability of high breakdown point; used to compute nsamp when nsamp is omitted. Defaults to 0.99.
eps	a numeric value between 0 and 0.5 specifying the breakdown point; used to compute nsamp when nresamp is omitted. Defaults to 0.5.
seed	starting value for random generator. Default is seed = NULL.
trace	whether to print intermediate results. Default is trace = FALSE.
control	a control object (S4) of class CovControlSde-class containing estimation options - same as these provided in the fucntion specification. If the control object is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object.

Details

The projection based Stahel-Donoho estimator posses very good statistical properties, but it can be very slow if the number of variables is too large. It is recommended to use this estimator if n <= 1000 and p<=10 or n <= 5000 and p<=5. The number of subsamples required is calculated to provide a breakdown point of eps with probability prob and can reach values larger than the larger integer value - in such case it is limited to .Machine$integer.max. Of course you could provide nsamp in the call, i.e. nsamp=1000 but this will not guarantee the required breakdown point of th eestimator. For larger data sets it is better to use CovMcd or CovOgk. If you use CovRobust, the estimator will be selected automatically according on the size of the data set.

Value

An S4 object of class CovSde-class which is a subclass of the virtual class CovRobust-class.

Note

The Fortran code for the Stahel-Donoho method was taken almost with no changes from package robust which in turn has it from the *Insightful Robust Library* (thanks to by Kjell Konis).

Author(s)

Valentin Todorov <valentin.todorov@chello.at> and Kjell Konis <kjell.konis@epfl.ch>

References

R. A. Maronna and V.J. Yohai (1995) The Behavior of the Stahel-Donoho Robust Multivariate Estimator. *Journal of the American Statistical Association* **90** (429), 330–341.

R. A. Maronna, D. Martin and V. Yohai (2006). *Robust Statistics: Theory and Methods*. Wiley, New York.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
CovSde(hbk.x)

## the following four statements are equivalent
c0 <- CovSde(hbk.x)
c1 <- CovSde(hbk.x, nsamp=2000)
c2 <- CovSde(hbk.x, control = CovControlSde(nsamp=2000))
c3 <- CovSde(hbk.x, control = new("CovControlSde", nsamp=2000))

## direct specification overrides control one:
c4 <- CovSde(hbk.x, nsamp=100,
             control = CovControlSde(nsamp=2000))
c1
summary(c1)
plot(c1)

## Use the function CovRobust() - if no estimation method is
##  specified, for small data sets CovSde() will be called
cr <- CovRobust(hbk.x)
cr
```

CovSde-class *Stahel-Donoho Estimates of Multivariate Location and Scatter*

Description

This class, derived from the virtual class "CovRobust" accomodates Stahel-Donoho estimates of multivariate location and scatter.

Objects from the Class

Objects can be created by calls of the form new("CovSde",...), but the usual way of creating CovSde objects is a call to the function CovSde which serves as a constructor.

Slots

iter, crit, wt: from the "CovRobust" class.

call, cov, center, n.obs, mah, method, singularity, X: from the "Cov" class.

Extends

Class "CovRobust", directly. Class "Cov", by class "CovRobust".

Methods

No methods defined with class "CovSde" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

CovSde, Cov-class, CovRobust-class

Examples

```
showClass("CovSde")
```

CovSest *S Estimates of Multivariate Location and Scatter*

Description

Computes S-Estimates of multivariate location and scatter based on Tukey's biweight function using a fast algorithm similar to the one proposed by Salibian-Barrera and Yohai (2006) for the case of regression. Alternativley, the Ruppert's SURREAL algorithm, bisquare or Rocke type estimation can be used.

Usage

```
CovSest(x, bdp = 0.5, arp = 0.1, eps = 1e-5, maxiter = 120,
    nsamp = 500, seed = NULL, trace = FALSE, tolSolve = 1e-14,
    scalefn, maxisteps=200, initHsets = NULL, save.hsets = FALSE,
  method = c("sfast", "surreal", "bisquare", "rocke", "suser", "sdet"), control,
    t0, S0, initcontrol)
```

Arguments

x	a matrix or data frame.
bdp	a numeric value specifying the required breakdown point. Allowed values are between (n - p)/(2 * n) and 1 and the default is bdp=0.5.
arp	a numeric value specifying the asympthotic rejection point (for the Rocke type S estimates), i.e. the fraction of points receiving zero weight (see Rocke (1996)). Default is arp=0.1.
eps	a numeric value specifying the relative precision of the solution of the S-estimate (bisquare and Rocke type). Default is to eps=1e-5.
maxiter	maximum number of iterations allowed in the computation of the S-estimate (bisquare and Rocke type). Default is maxiter=120.

nsamp	the number of random subsets considered. The default is different for the different methods: (i) for sfast it is nsamp = 20, (ii) for surreal it is nsamp = 600*p and (iii) for bisquare or rocke it is nsamp = 500.
seed	starting value for random generator. Default is seed = NULL.
trace	whether to print intermediate results. Default is trace = FALSE.
tolSolve	numeric tolerance to be used for inversion (solve) of the covariance matrix in mahalanobis.
scalefn	function to compute a robust scale estimate or character string specifying a rule determining such a function. Used for computing the "deterministic" S-estimates (method="sdet"). If scalefn is missing or is NULL, the function is selected depending on the data set size, following the recomendation of Hubert et al. (2012) - Qn if n <= 1000 and scaleTau2 otherwise.
maxisteps	maximal number of concentration steps in the deterministic S-estimates; should not be reached.
initHsets	NULL or a Kxn integer matrix of initial subsets of observations of size (specified by the indices in 1:n).
save.hsets	(for deterministic S-estimates) logical indicating if the initial subsets should be returned as initHsets.
method	Which algorithm to use: 'sfast'=C implementation of FAST-S, 'surreal'=SURREAL, 'bisquare', 'rocke'. The method 'suser' currently calls the R implementation of FAST-S but in the future will allow the user to supply own rho function. The method 'sdet' invokes the deterministic algorihm of Hubert et al. (2012).
control	a control object (S4) of class CovControlSest-class containing estimation options - same as these provided in the fucntion specification. If the control object is supplied, the parameters from it will be used. If parameters are passed also in the invocation statement, they will override the corresponding elements of the control object.
t0	optional initial HBDP estimate for the center
S0	optional initial HBDP estimate for the covariance matrix
initcontrol	optional control object to be used for computing the initial HBDP estimates

Details

Computes multivariate S-estimator of location and scatter. The computation will be performed by one of the following algorithms:

FAST-S An algorithm similar to the one proposed by Salibian-Barrera and Yohai (2006) for the case of regression

SURREAL Ruppert's SURREAL algorithm when method is set to 'surreal'

BISQUARE Bisquare S-Estimate with method set to 'bisquare'

ROCKE Rocke type S-Estimate with method set to 'rocke'

Except for the last algorithm, *ROCKE*, all other use Tukey biweight loss function. The tuning parameters used in the loss function (as determined by bdp) are returned in the slots cc and kp of the result object. They can be computed by the internal function .csolve.bw.S(bdp,p).

Value

An S4 object of class CovSest-class which is a subclass of the virtual class CovRobust-class.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>, Matias Salibian-Barrera <matias@stat.ubc.ca> and Victor Yohai <vyohai@dm.uba.ar>. See also the code from Kristel Joossens, K.U. Leuven, Belgium and Ella Roelant, Ghent University, Belgium.

References

M. Hubert, P. Rousseeuw and T. Verdonck (2012) A deterministic algorithm for robust location and scatter. *Journal of Computational and Graphical Statistics* **21**(3), 618–637.

M. Hubert, P. Rousseeuw, D. Vanpaemel and T. Verdonck (2015) The DetS and DetMM estimators for multivariate location and scatter. *Computational Statistics and Data Analysis* **81**, 64–75.

H.P. Lopuhaä (1989) On the Relation between S-estimators and M-estimators of Multivariate Location and Covariance. *Annals of Statistics* **17** 1662–1683.

D. Ruppert (1992) Computing S Estimators for Regression and Multivariate Location/Dispersion. *Journal of Computational and Graphical Statistics* **1** 253–270.

M. Salibian-Barrera and V. Yohai (2006) A fast algorithm for S-regression estimates, *Journal of Computational and Graphical Statistics*, **15**, 414–427.

R. A. Maronna, D. Martin and V. Yohai (2006). *Robust Statistics: Theory and Methods*. Wiley, New York.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
library(rrcov)
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
cc <- CovSest(hbk.x)
cc

## summry and different types of plots
summary(cc)
plot(cc)
plot(cc, which="dd")
plot(cc, which="pairs")
plot(cc, which="xydist")

## the following four statements are equivalent
c0 <- CovSest(hbk.x)
c1 <- CovSest(hbk.x, bdp = 0.25)
c2 <- CovSest(hbk.x, control = CovControlSest(bdp = 0.25))
c3 <- CovSest(hbk.x, control = new("CovControlSest", bdp = 0.25))
```

```
## direct specification overrides control one:
c4 <- CovSest(hbk.x, bdp = 0.40,
             control = CovControlSest(bdp = 0.25))
c1
summary(c1)
plot(c1)

## Use the SURREAL algorithm of Ruppert
cr <- CovSest(hbk.x, method="surreal")
cr

## Use Bisquare estimation
cr <- CovSest(hbk.x, method="bisquare")
cr

## Use Rocke type estimation
cr <- CovSest(hbk.x, method="rocke")
cr

## Use Deterministic estimation
cr <- CovSest(hbk.x, method="sdet")
cr
```

CovSest-class *S Estimates of Multivariate Location and Scatter*

Description

This class, derived from the virtual class "CovRobust" accomodates S Estimates of multivariate location and scatter computed by the 'Fast S' or 'SURREAL' algorithm.

Objects from the Class

Objects can be created by calls of the form `new("CovSest",...)`, but the usual way of creating `CovSest` objects is a call to the function `CovSest` which serves as a constructor.

Slots

`iter`, `crit`, `wt`: from the "CovRobust" class.

`iBest`, `nsteps`, `initHsets`: parameters for deterministic S-estimator (the best initial subset, number of concentration steps to convergence for each of the initial subsets, and the computed initial subsets, respectively).

`cc`, `kp` tuning parameters used in Tukey biweight loss function, as determined by bdp. Can be computed by the internal function `.csolve.bw.S(bdp,p)`.

`call`, `cov`, `center`, `n.obs`, `mah`, `method`, `singularity`, `X`: from the "Cov" class.

Extends

Class "CovRobust", directly. Class "Cov", by class "CovRobust".

Methods

No methods defined with class "CovSest" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

CovSest, Cov-class, CovRobust-class

Examples

```
showClass("CovSest")
```

---

diabetes — *Reaven and Miller diabetes data*

Description

The data set contains five measurements made on 145 non-obese adult patients classified into three groups.

The three primary variables are glucose intolerance (area under the straight line connecting glucose levels), insulin response to oral glucose (area under the straight line connecting insulin levels) and insulin resistance (measured by the steady state plasma glucose (SSPG) determined after chemical suppression of endogenous insulin secretion). Two additional variables, the relative weight and fasting plasma glucose, are also included.

Reaven and Miller, following Friedman and Rubin (1967), applied cluster analysis to the three primary variables and identified three clusters: "normal", "chemical diabetic", and "overt diabetic" subjects. The column `group` contains the classifications of the subjects into these three groups, obtained by current medical criteria.

Usage

```
data(diabetes)
```

Format

A data frame with the following variables:

rw relative weight, expressed as the ratio of actual weight to expected weight, given the person's height.

fpg fasting plasma glucose level.

glucose area under plasma glucose curve after a three hour oral glucose tolerance test (OGTT).

insulin area under plasma insulin curve after a three hour oral glucose tolerance test (OGTT).

sspg Steady state plasma glucose, a measure of insulin resistance.

group the type of diabetes: a factor with levels `normal`, `chemical` and `overt`.

Source

Reaven, G. M. and Miller, R. G. (1979). An attempt to define the nature of chemical diabetes using a multidimensional analysis. *Diabetologia* 16, 17–24. Andrews, D. F. and Herzberg, A. M. (1985). *Data: A Collection of Problems from Many Fields for the Student and Research Worker*, Springer-Verlag, Ch. 36.

References

Reaven, G. M. and Miller, R. G. (1979). An attempt to define the nature of chemical diabetes using a multidimensional analysis. *Diabetologia* 16, 17–24.

Friedman, H. P. and Rubin, J. (1967). On some invariant criteria for grouping data. *Journal of the American Statistical Association* **62**, 1159–1178.

Hawkins, D. M. and McLachlan, G. J., 1997. High-breakdown linear discriminant analysis. *Journal of the American Statistical Association* **92** (437), 136–143.

Examples

```
data(diabetes)
(cc <- Linda(group~insulin+glucose+sspg, data=diabetes))
(pr <- predict(cc))
```

`fish` *Fish Catch Data Set*

Description

The Fish Catch data set contains measurements on 159 fish caught in the lake Laengelmavesi, Finland.

Usage

```
data(fish)
```

Format

A data frame with 159 observations on the following 7 variables.

`Weight` Weight of the fish (in grams)

`Length1` Length from the nose to the beginning of the tail (in cm)

`Length2` Length from the nose to the notch of the tail (in cm)

`Length3` Length from the nose to the end of the tail (in cm)

`Height` Maximal height as % of Length3

`Width` Maximal width as % of Length3

`Species` Species

Details

The Fish Catch data set contains measurements on 159 fish caught in the lake Laengelmavesi, Finland. For the 159 fishes of 7 species the weight, length, height, and width were measured. Three different length measurements are recorded: from the nose of the fish to the beginning of its tail, from the nose to the notch of its tail and from the nose to the end of its tail. The height and width are calculated as percentages of the third length variable. This results in 6 observed variables, Weight, Length1, Length2, Length3, Height, Width. Observation 14 has a missing value in variable Weight, therefore this observation is usually excluded from the analysis. The last variable, Species, represents the grouping structure: the 7 species are 1=Bream, 2=Whitewish, 3=Roach, 4=Parkki, 5=Smelt, 6=Pike, 7=Perch. This data set was also analyzed in the context of robust Linear Discriminant Analysis by Todorov (2007), Todorov and Pires (2007).

Source

Journal of Statistical Education, Fish Catch Data Set, [http://www.amstat.org/publications/jse/datasets/fishcatch.txt] accessed August, 2006.

References

Todorov, V. (2007 Robust selection of variables in linear discriminant analysis, *Statistical Methods and Applications*, **15**, 395–407, doi:10.1007/s10260-006-0032-6.

Todorov, V. and Pires, A.M. (2007) Comparative performance of several robust linear discriminant analysis methods, *REVSTAT Statistical Journal*, **5**, 63–83.

Examples

```
data(fish)

# remove observation #14 containing missing value
fish <- fish[-14,]

# The height and width are calculated as percentages
#   of the third length variable
fish[,5] <- fish[,5]*fish[,4]/100
fish[,6] <- fish[,6]*fish[,4]/100
```

```
# plot a matrix of scatterplots
pairs(fish[1:6],
      main="Fish Catch Data",
      pch=21,
      bg=c("red", "green3", "blue", "yellow", "magenta", "violet",
      "turquoise")[unclass(fish$Species)])
```

---

getCenter-methods *Accessor methods to the essential slots of Cov and its subclasses*

Description

Accessor methods to the slots of objects of classCov and its subclasses

Usage

```
getCenter(obj)
getCov(obj)
getCorr(obj)
getData(obj)
getDistance(obj)
getEvals(obj)
getDet(obj)
getShape(obj)
getFlag(obj, prob=0.975)
getMeth(obj)
isClassic(obj)
getRaw(obj)
```

Arguments

obj	an object of class "Cov" or of a class derived from "Cov"
prob	optional argument for getFlag - probability, defaults to 0.975

Methods

obj = "Cov" generic functions - see getCenter, getCov, getCorr, getData, getDistance, getEvals, getDet, getShape, getFlag, isClassic

obj = "CovRobust" generic functions - see getCenter, getCov, getCorr, getData, getDistance, getEvals, getDet, getShape, getFlag, getMeth, isClassic `getEllipse` *Calculates the points for drawing a confidence ellipsoid*

Description

A simple function to calculate the points of a confidence ellipsoid, by default `dist=qchisq(0.975,2)`

Usage

```
getEllipse(loc = c(0, 0), cov = matrix(c(1, 0, 0, 1), ncol = 2), crit = 0.975)
```

Arguments

`loc`	location vector
`cov`	a pXp covariance matrix
`crit`	the confidence level, default is `crit=0.975`

Value

A matrix with two columns containing the calculated points.

Author(s)

Valentin Todorov, `<valentin.todorov@chello.at>`

Examples

```
data(hbk)
cc <- cov.wt(hbk)
e1 <- getEllipse(loc=cc$center[1:2], cov=cc$cov[1:2,1:2])
e2 <- getEllipse(loc=cc$center[1:2], cov=cc$cov[1:2,1:2], crit=0.99)
plot(X2~X1, data=hbk,
    xlim=c(min(X1, e1[,1], e2[,1]), max(X1,e1[,1], e2[,1])),
    ylim=c(min(X2, e1[,2], e2[,2]), max(X2,e1[,2], e2[,2])))
lines(e1, type="l", lty=1, col="red")
lines(e2, type="l", lty=2, col="blue")
legend("topleft", legend=c(0.975, 0.99), lty=1:2, col=c("red", "blue"))
```

getLoadings-methods *Accessor methods to the essential slots of Pca and its subclasses*

Description

Accessor methods to the slots of objects of class `Pca` and its subclasses

Arguments

`obj` an object of class `"Pca"` or of a class derived from `"Pca"`

Methods

obj = "Pca" Accessors for object of class `Pca`

obj = "PcaRobust" Accessors for object of class `PcaRobust`

obj = "PcaClassic" Accessors for object of class `PcaClassic`

hemophilia *Hemophilia Data*

Description

The hemophilia data set contains two measured variables on 75 women, belonging to two groups: n1=30 of them are non-carriers (normal group) and n2=45 are known hemophilia A carriers (obligatory carriers).

Usage

```
data(hemophilia)
```

Format

A data frame with 75 observations on the following 3 variables.

`AHFactivity` AHF activity

`AHFantigen` AHF antigen

`gr` group - normal or obligatory carrier

Details

Originally analized in the context of discriminant analysis by Habemma and Hermans (1974). The objective is to find a procedure for detecting potential hemophilia A carriers on the basis of two measured variables: X1=log10(AHV activity) and X2=log10(AHV-like antigen). The first group of n1=30 women consists of known non-carriers (normal group) and the second group of n2=45 women is selected from known hemophilia A carriers (obligatory carriers). This data set was also analyzed by Johnson and Wichern (1998) as well as, in the context of robust Linear Discriminant Analysis by Hawkins and McLachlan (1997) and Hubert and Van Driessen (2004).

Source

Habemma, J.D.F, Hermans, J. and van den Broek, K. (1974) Stepwise Discriminant Analysis Program Using Density Estimation in *Proceedings in Computational statistics, COMPSTAT'1974* (Physica Verlag, Heidelberg, 1974, pp 101–110).

References

Johnson, R.A. and Wichern, D. W. *Applied Multivariate Statistical Analysis* (Prentice Hall, International Editions, 2002, fifth edition)

Hawkins, D. M. and McLachlan, G.J. (1997) High-Breakdown Linear Discriminant Analysis *J. Amer. Statist. Assoc.* **92** 136–143.

Hubert, M., Van Driessen, K. (2004) Fast and robust discriminant analysis, *Computational Statistics and Data Analysis*, **45** 301–320.

Examples

```
data(hemophilia)
plot(AHFantigen~AHFactivity, data=hemophilia, col=as.numeric(as.factor(gr))+1)
##
## Compute robust location and covariance matrix and
## plot the tolerance ellipses
(mcd <- CovMcd(hemophilia[,1:2]))
col <- ifelse(hemophilia$gr == "carrier", 2, 3) ## define clours for the groups
plot(mcd, which="tolEllipsePlot", class=TRUE, col=col)
```

---

isSingular-methods *Check if a covariance matrix (object of class 'Cov') is singular*

Description

Returns TRUE if the covariance matrix contained in a `Cov-class` object (or derived from) is singular.

Usage

```
    ## S4 method for signature 'Cov'
isSingular(obj)
```

Arguments

`obj`	an object of class (derived from) "Cov".

Methods

isSingular `signature(x = Cov)`: Check if a covariance matrix (object of class `Cov-class`) is singular.

See Also

`Cov-class`, `CovClassic`, `CovRobust-class`.

Examples

```
data(hbk)
cc <- CovClassic(hbk)
isSingular(cc)
```

`Lda-class` *Class "Lda" - virtual base class for all classic and robust LDA classes*

Description

The class `Lda` serves as a base class for deriving all other classes representing the results of classical and robust Linear Discriminant Analisys methods

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`call`: the (matched) function call.

`prior`: prior probabilities used, default to group proportions

`counts`: number of observations in each class

`center`: the group means

`cov`: the common covariance matrix

`ldf`: a matrix containing the linear discriminant functions

`ldfconst`: a vector containing the constants of each linear discriminant function

`method`: a character string giving the estimation method used

`X`: the training data set (same as the input parameter x of the constructor function)

`grp`: grouping variable: a factor specifying the class for each observation.

`covobj`: object of class `"Cov"` containing the estimate of the common covariance matrix of the centered data. It is not NULL only in case of method "B".

`control`: object of class `"CovControl"` specifying which estimate and with what estimation options to use for the group means and common covariance (or `NULL` for classical linear discriminant analysis)

Methods

predict `signature(object = "Lda")`: calculates prediction using the results in `object`. An optional data frame or matrix in which to look for variables with which to predict. If omitted, the training data set is used. If the original fit used a formula or a data frame or a matrix with column names, newdata must contain columns with the same names. Otherwise it must contain the same number of columns, to be used in the same order.

show `signature(object = "Lda")`: prints the results

summary `signature(object = "Lda")`: prints summary information

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

LdaClassic, LdaClassic-class, LdaRobust-class

Examples

```
showClass("Lda")
```

LdaClassic *Linear Discriminant Analysis*

Description

Performs a linear discriminant analysis and returns the results as an object of class `LdaClassic` (aka constructor).

Usage

```
LdaClassic(x, ...)


## Default S3 method:
LdaClassic(x, grouping, prior = proportions, tol = 1.0e-4, ...)
```

Arguments

x	a matrix or data frame containing the explanatory variables (training set).
grouping	grouping variable: a factor specifying the class for each observation.
prior	prior probabilities, default to the class proportions for the training set.
tol	tolerance
...	arguments passed to or from other methods.

Value

Returns an S4 object of class `LdaClassic`

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

`Lda-class`, `LdaClassic-class`,

Examples

```
## Example anorexia
library(MASS)
data(anorexia)

## rrcov: LdaClassic()
lda <- LdaClassic(Treat~., data=anorexia)
predict(lda)@classification

## MASS: lda()
lda.MASS <- lda(Treat~., data=anorexia)
predict(lda.MASS)$class

## Compare the prediction results of MASS:::lda() and LdaClassic()
all.equal(predict(lda)@classification, predict(lda.MASS)$class)
```

LdaClassic-class *Class "LdaClassic" - Linear Discriminant Analysis*

Description

Contains the results of a classical Linear Discriminant Analysis

Objects from the Class

Objects can be created by calls of the form `new("LdaClassic",...)` but the usual way of creating `LdaClassic` objects is a call to the function `LdaClassic` which serves as a constructor.

Slots

`call`**:** The (matched) function call.

`prior`**:** Prior probabilities used, default to group proportions

`counts`**:** number of observations in each class

`center`**:** the group means

`cov`**:** the common covariance matrix

`ldf`**:** a matrix containing the linear discriminant functions

`ldfconst`**:** a vector containing the constants of each linear discriminant function

`method`**:** a character string giving the estimation method used

`X`**:** the training data set (same as the input parameter x of the constructor function)

`grp`**:** grouping variable: a factor specifying the class for each observation.

Extends

Class "`Lda`", directly.

Methods

No methods defined with class "LdaClassic" in the signature.

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

See Also

`LdaRobust-class`, `Lda-class`, `LdaClassic`

Examples

```
showClass("LdaClassic")
```

LdaPP *Robust Linear Discriminant Analysis by Projection Pursuit*

Description

Performs robust linear discriminant analysis by the projection-pursuit approach - proposed by Pires and Branco (2010) - and returns the results as an object of class LdaPP (aka constructor).

Usage

```
LdaPP(x, ...)
## S3 method for class 'formula'
LdaPP(formula, data, subset, na.action, ...)
## Default S3 method:
LdaPP(x, grouping, prior = proportions, tol = 1.0e-4,
                 method = c("huber", "mad", "sest", "class"),
                 optim = FALSE,
                 trace=FALSE, ...)
```

Arguments

formula	a formula of the form y~x, it describes the response and the predictors. The formula can be more complicated, such as y~log(x)+z etc (see formula for more details). The response should be a factor representing the response variable, or any vector that can be coerced to such (such as a logical variable).
data	an optional data frame (or similar: see model.frame) containing the variables in the formula formula.
subset	an optional vector used to select rows (observations) of the data matrix x.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The default is na.omit.
x	a matrix or data frame containing the explanatory variables (training set).
grouping	grouping variable: a factor specifying the class for each observation.
prior	prior probabilities, default to the class proportions for the training set.
tol	tolerance
method	method
optim	wheather to perform the approximation using the Nelder and Mead simplex method (see function optim() from package stats). Default is optim = FALSE
trace	whether to print intermediate results. Default is trace = FALSE.
...	arguments passed to or from other methods.

Details

Currently the algorithm is implemented only for binary classification and in the following will be assumed that only two groups are present.

The PP algorithm searches for low-dimensional projections of higher-dimensional data where a projection index is maximized. Similar to the original Fisher's proposal the squared standardized distance between the observations in the two groups is maximized. Instead of the sample univariate mean and standard deviation `(T,S)` robust alternatives are used. These are selected through the argument `method` and can be one of

huber the pair `(T,S)` are the robust M-estimates of location and scale

mad `(T,S)` are the Median and the Median Absolute Deviation

sest the pair `(T,S)` are the robust S-estimates of location and scale

class `(T,S)` are the mean and the standard deviation.

The first approximation *A1* to the solution is obtained by investigating a finite number of candidate directions, the unit vectors defined by all pairs of points such that one belongs to the first group and the other to the second group. The found solution is stored in the slots `raw.ldf` and `raw.ldfconst`.

The second approximation *A2* (optional) is performed by a numerical optimization algorithm using *A1* as initial solution. The Nelder and Mead method implemented in the function `optim` is applied. Whether this refinement will be used is controlled by the argument `optim`. If `optim=TRUE` the result of the optimization is stored into the slots `ldf` and `ldfconst`. Otherwise these slots are set equal to `raw.ldf` and `raw.ldfconst`.

Value

Returns an S4 object of class `LdaPP-class`

Warning

Still an experimental version! Only binary classification is supported.

Author(s)

Valentin Todorov <valentin.todorov@chello.at> and Ana Pires <apires@math.ist.utl.pt>

References

Pires, A. M. and A. Branco, J. (2010) Projection-pursuit approach to robust linear discriminant analysis *Journal Multivariate Analysis*, Academic Press, Inc., **101**, 2464–2485.

See Also

`Linda`, `LdaClassic`

Examples

```
##
## Function to plot a LDA separation line
##
lda.line <- function(lda, ...)
{
    ab <- lda@ldf[1,] - lda@ldf[2,]
    cc <- lda@ldfconst[1] - lda@ldfconst[2]
    abline(a=-cc/ab[2], b=-ab[1]/ab[2],...)
}

data(pottery)
x <- pottery[,c("MG", "CA")]
grp <- pottery$origin
col <- c(3,4)
gcol <- ifelse(grp == "Attic", col[1], col[2])
gpch <- ifelse(grp == "Attic", 16, 1)

##
## Reproduce Fig. 2. from Pires and branco (2010)
##
plot(CA~MG, data=pottery, col=gcol, pch=gpch)

ppc <- LdaPP(x, grp, method="class", optim=TRUE)
lda.line(ppc, col=1, lwd=2, lty=1)

pph <- LdaPP(x, grp, method="huber",optim=TRUE)
lda.line(pph, col=3, lty=3)

pps <- LdaPP(x, grp, method="sest", optim=TRUE)
lda.line(pps, col=4, lty=4)

ppm <- LdaPP(x, grp, method="mad", optim=TRUE)
lda.line(ppm, col=5, lty=5)

rlda <- Linda(x, grp, method="mcd")
lda.line(rlda, col=6, lty=1)

fsa <- Linda(x, grp, method="fsa")
lda.line(fsa, col=8, lty=6)

## Use the formula interface:
##
LdaPP(origin~MG+CA, data=pottery)       ## use the same two predictors
LdaPP(origin~., data=pottery)           ## use all predictor variables

##
## Predict method
data(pottery)
fit <- LdaPP(origin~., data = pottery)
predict(fit)
```

LdaPP-class *Class "LdaPP" - Robust method for Linear Discriminant Analysis by Projection-pursuit*

Description

The class LdaPP represents an algorithm for robust linear discriminant analysis by projection-pursuit approach. The objects of class LdaPP contain the results of the robust linear discriminant analysis by projection-pursuit approach.

Objects from the Class

Objects can be created by calls of the form new("LdaPP",...) but the usual way of creating LdaPP objects is a call to the function LdaPP which serves as a constructor.

Slots

call: The (matched) function call.

prior: Prior probabilities used, default to group proportions

counts: number of observations in each class

center: the group means

cov: the common covariance matrix

raw.ldf: a matrix containing the raw linear discriminant functions - see Details in LdaPP

raw.ldfconst: a vector containing the raw constants of each raw linear discriminant function - see Details in LdaPP

ldf: a matrix containing the linear discriminant functions

ldfconst: a vector containing the constants of each linear discriminant function

method: a character string giving the estimation method used

X: the training data set (same as the input parameter x of the constructor function)

grp: grouping variable: a factor specifying the class for each observation.

Extends

Class "LdaRobust", directly. Class "Lda", by class "LdaRobust", distance 2.

Methods

predict signature(object = "LdaPP"): calculates prediction using the results in object. An optional data frame or matrix in which to look for variables with which to predict. If omitted, the training data set is used. If the original fit used a formula or a data frame or a matrix with column names, newdata must contain columns with the same names. Otherwise it must contain the same number of columns, to be used in the same order. If the argument raw=TRUE is set the raw (obtained by the first approximation algorithm) linear discriminant function and constant will be used.

Author(s)

Valentin Todorov <valentin.todorov@chello.at> and Ana Pires <apires@math.ist.utl.pt>

References

Pires, A. M. and A. Branco, J. (2010) Projection-pursuit approach to robust linear discriminant analysis *Journal Multivariate Analysis*, Academic Press, Inc., **101**, 2464–2485.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

LdaRobust-class, Lda-class, LdaClassic, LdaClassic-class, Linda, Linda-class

Examples

```
showClass("LdaPP")
```

LdaRobust-class — *Class "LdaRobust" is a virtual base class for all robust LDA classes*

Description

The class `LdaRobust` searves as a base class for deriving all other classes representing the results of the robust Linear Discriminant Analysis methods

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`call`**:** The (matched) function call.

`prior`**:** Prior probabilities used, default to group proportions

`counts`**:** number of observations in each class

`center`**:** the group means

`cov`**:** the common covariance matrix

`ldf`**:** a matrix containing the linear discriminant functions

`ldfconst`**:** a vector containing the constants of each linear discriminant function

`method`**:** a character string giving the estimation method used

`X`**:** the training data set (same as the input parameter x of the constructor function)

`grp`**:** grouping variable: a factor specifying the class for each observation.

Extends

Class "Lda", directly.

Methods

No methods defined with class "LdaRobust" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

Lda-class, LdaClassic-class,

Examples

```
showClass("LdaRobust")
```

Linda *Robust Linear Discriminant Analysis*

Description

Robust linear discriminant analysis based on MCD and returns the results as an object of class Linda (aka constructor).

Usage

```
Linda(x, ...)

## Default S3 method:
Linda(x, grouping, prior = proportions, tol = 1.0e-4,
                method = c("mcd", "mcdA", "mcdB", "mcdC", "fsa", "mrcd", "ogk"),
                 alpha=0.5, l1med=FALSE, cov.control, trace=FALSE, ...)
```

Arguments

`x`	a matrix or data frame containing the explanatory variables (training set).
`grouping`	grouping variable: a factor specifying the class for each observation.
`prior`	prior probabilities, default to the class proportions for the training set.
`tol`	tolerance
`method`	method
`alpha`	this parameter measures the fraction of outliers the algorithm should resist. In MCD alpha controls the size of the subsets over which the determinant is minimized, i.e. alpha*n observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.5.
`l1med`	whether to use L1 median (space median) instead of MCD to compute the group means locations in order to center the data in methods `mcdB` and `mcdC`. useful in case of groups with small size. Default is `l1med = FALSE`.
`cov.control`	specifies which covariance estimator to use by providing a `CovControl-class` object. The default is `CovControlMcd-class` which will indirectly call `CovMcd`. If `cov.control=NULL` is specified, the classical estimates will be used by calling `CovClassic`.
`trace`	whether to print intermediate results. Default is `trace = FALSE`.
`...`	arguments passed to or from other methods

Details

details

Value

Returns an S4 object of class `Linda`

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Hawkins, D.M. and McLachlan, G.J. (1997) High-Breakdown Linear Discriminant Analysis, *Journal of the American Statistical Association*, **92**, 136–143.

Todorov V. (2007) Robust selection of variables in linear discriminant analysis, *Statistical Methods and Applications*, **15**, 395–407, doi:10.1007/s10260-006-0032-6.

Todorov, V. and Pires, A.M. (2007) Comparative Performance of Several Robust Linear Discriminant Analysis Methods. *REVSTAT Statistical Journal*, **5**, p 63–83. URL `http://www.ine.pt/revstat/pdf/rs070104.pdf`.

Todorov V and Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

See Also

CovMcd, CovMrcd

Examples

```
## Example anorexia
library(MASS)
data(anorexia)

## start with the classical estimates
lda <- LdaClassic(Treat~., data=anorexia)
predict(lda)@classification

## try now the robust LDA with the default method (MCD with pooled whitin cov matrix)
rlda <- Linda(Treat~., data= anorexia)
predict(rlda)@classification

## try the other methods
Linda(Treat~., data= anorexia, method="mcdA")
Linda(Treat~., data= anorexia, method="mcdB")
Linda(Treat~., data= anorexia, method="mcdC")

## try the Hawkins&McLachlan method
## use the default method
grp <- anorexia[,1]
grp <- as.factor(grp)
x <- anorexia[,2:3]
Linda(x, grp, method="fsa")

## Do DA with Linda and method mcdB or mcdC, when some classes
## have very few observations. Use L1 median instead of MCD
##  to compute the group means (l1med=TRUE).

data(fish)

# remove observation #14 containing missing value
fish <- fish[-14,]

# The height and width are calculated as percentages
#   of the third length variable
fish[,5] <- fish[,5]*fish[,4]/100
fish[,6] <- fish[,6]*fish[,4]/100

table(fish$Species)
Linda(Species~., data=fish, l1med=TRUE)
Linda(Species~., data=fish, method="mcdC", l1med=TRUE)
```

Linda-class *Class "Linda" - Robust method for LINear Discriminant Analysis*

Description

Robust linear discriminant analysis is performed by replacing the classical group means and withing group covariance matrix by robust equivalents based on MCD.

Objects from the Class

Objects can be created by calls of the form `new("Linda",...)` but the usual way of creating `Linda` objects is a call to the function `Linda` which serves as a constructor.

Slots

`call`: The (matched) function call.

`prior`: Prior probabilities used, default to group proportions

`counts`: number of observations in each class

`center`: the group means

`cov`: the common covariance matrix

`ldf`: a matrix containing the linear discriminant functions

`ldfconst`: a vector containing the constants of each linear discriminant function

`method`: a character string giving the estimation method used

`X`: the training data set (same as the input parameter x of the constructor function)

`grp`: grouping variable: a factor specifying the class for each observation.

`l1med`: wheather L1 median was used to compute group means.

Extends

Class "`LdaRobust`", directly. Class "`Lda`", by class "LdaRobust", distance 2.

Methods

No methods defined with class "Linda" in the signature.

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

See Also

`LdaRobust-class`, `Lda-class`, `LdaClassic`, `LdaClassic-class`

Examples

```
showClass("Linda")
```

lmom32 *Hosking and Wallis Data Set, Table 3.2*

Description

The data on annual maximum streamflow at 18 sites with smallest drainage area basin in south-eastern USA contains the sample L-moments ratios (L-CV, L-skewness and L-kurtosis) as used by Hosking and Wallis (1997) to illustrate the discordancy measure in regional freqency analysis (RFA).

Usage

```
data(lmom32)
```

Format

A data frame with 18 observations on the following 3 variables.

`L-CV` L-coefficient of variation

`L-skewness` L-coefficient of skewness

`L-kurtosis` L-coefficient of kurtosis

Details

The sample L-moment ratios (L-CV, L-skewness and L-kurtosis) of a site are regarded as a point in three dimensional space.

Source

Hosking, J. R. M. and J. R. Wallis (1997), *Regional Frequency Analysis: An Approach Based on L-moments.* Cambridge University Press, p.49, Table 3.2

References

Neykov, N.M., Neytchev, P.N., Van Gelder, P.H.A.J.M. and Todorov V. (2007), Robust detection of discordant sites in regional frequency analysis, *Water Resources Research*, 43, W06417, doi:10.1029/2006WR005322

Examples

```
data(lmom32)

# plot a matrix of scatterplots
pairs(lmom32,
      main="Hosking and Wallis Data Set, Table 3.3",
      pch=21,
      bg=c("red", "green3", "blue"))
```

```
mcd<-CovMcd(lmom32)
mcd
plot(mcd, which="dist", class=TRUE)
plot(mcd, which="dd", class=TRUE)

##  identify the discordant sites using robust distances and compare
##  to the classical ones
mcd <- CovMcd(lmom32)
rd <- sqrt(getDistance(mcd))
ccov <- CovClassic(lmom32)
cd <- sqrt(getDistance(ccov))
r.out <- which(rd > sqrt(qchisq(0.975,3)))
c.out <- which(cd > sqrt(qchisq(0.975,3)))
cat("Robust: ", length(r.out), " outliers: ", r.out,"\n")
cat("Classical: ", length(c.out), " outliers: ", c.out,"\n")
```

`lmom33` *Hosking and Wallis Data Set, Table 3.3*

Description

The data on annual maximum streamflow at 17 sites with largest drainage area basins in south-eastern USA contains the sample L-moments ratios (L-CV, L-skewness and L-kurtosis) as used by Hosking and Wallis (1997) to illustrate the discordancy measure in regional freqency analysis (RFA).

Usage

```
data(lmom33)
```

Format

A data frame with 17 observations on the following 3 variables.

`L-CV` L-coefficient of variation

`L-skewness` L-coefficient of skewness

`L-kurtosis` L-coefficient of kurtosis

Details

The sample L-moment ratios (L-CV, L-skewness and L-kurtosis) of a site are regarded as a point in three dimensional space.

Source

Hosking, J. R. M. and J. R. Wallis (1997), *Regional Frequency Analysis: An Approach Based on L-moments.* Cambridge University Press, p.51, Table 3.3

References

Neykov, N.M., Neytchev, P.N., Van Gelder, P.H.A.J.M. and Todorov V. (2007), Robust detection of discordant sites in regional frequency analysis, *Water Resources Research*, 43, W06417, doi:10.1029/2006WR005322

Examples

```
data(lmom33)

# plot a matrix of scatterplots
pairs(lmom33,
      main="Hosking and Wallis Data Set, Table 3.3",
      pch=21,
      bg=c("red", "green3", "blue"))

mcd<-CovMcd(lmom33)
mcd
plot(mcd, which="dist", class=TRUE)
plot(mcd, which="dd", class=TRUE)

##  identify the discordant sites using robust distances and compare
##  to the classical ones
mcd <- CovMcd(lmom33)
rd <- sqrt(getDistance(mcd))
ccov <- CovClassic(lmom33)
cd <- sqrt(getDistance(ccov))
r.out <- which(rd > sqrt(qchisq(0.975,3)))
c.out <- which(cd > sqrt(qchisq(0.975,3)))
cat("Robust: ", length(r.out), " outliers: ", r.out,"\n")
cat("Classical: ", length(c.out), " outliers: ", c.out,"\n")
```

---

maryo *Marona and Yohai Artificial Data*

Description

Simple artificial data set generated according the example by Marona and Yohai (1998). The data set consists of 20 bivariate normal observations generated with zero means, unit variances and correlation 0.8. The sample correlation is 0.81. Two outliers are introduced (i.e. these are 10% of the data) in the following way: two points are modified by interchanging the largest (observation 19) and smallest (observation 9) value of the first coordinate. The sample correlation becomes 0.05. This example provides a good example of the fact that a multivariate outlier need not be an outlier in any of its coordinate variables.

Usage

```
data(maryo)
```

Format

A data frame with 20 observations on 2 variables. To introduce the outliers x[9,1] with x[19,1] are interchanged.

Source

R. A. Marona and V. J. Yohai (1998) Robust estimation of multivariate location and scatter. In *Encyclopedia of Statistical Sciences, Updated Volume 2* (Eds. S.Kotz, C.Read and D.Banks). Wiley, New York p. 590

Examples

```
data(maryo)
getCorr(CovClassic(maryo))          ## the sample correlation is 0.81

## Modify 10%% of the data in the following way:
##  modify two points (out of 20) by interchanging the
##  largest and smallest value of the first coordinate
imin <- which(maryo[,1]==min(maryo[,1]))        # imin = 9
imax <- which(maryo[,1]==max(maryo[,1]))        # imax = 19
maryo1 <- maryo
maryo1[imin,1] <- maryo[imax,1]
maryo1[imax,1] <- maryo[imin,1]

##  The sample correlation becomes 0.05
plot(maryo1)
getCorr(CovClassic(maryo1))         ## the sample correlation becomes 0.05
getCorr(CovMcd(maryo1))      ## the (reweighted) MCD correlation is 0.79
```

`octane` *Octane data*

Description

The octane data contains near infrared absorbance spectra (NIR) of `n=39` gasoline samples over `p=226` wavelengths ranging from 1102 nm to 1552 nm with measurements every two nanometers. For each of the 39 production gasoline samples the octane number was measured. Six of the samples (25, 26, and 36-39) contain added alcohol.

Usage

```
data(octane)
```

Format

A data frame with 39 observations and 226 columns, the wavelengts are by column.

Source

K.H. Esbensen, S. Schoenkopf and T. Midtgaard *Multivariate Analysis in Practice*, Trondheim, Norway: Camo, 1994.

References

M. Hubert, P. J. Rousseeuw, K. Vanden Branden (2005), ROBPCA: a new approach to robust principal components analysis, *Technometrics*, **47**, 64–79.

P. J. Rousseeuw, M. Debruyne, S. Engelen and M. Hubert (2006), Robustness and Outlier Detection in Chemometrics, *Critical Reviews in Analytical Chemistry*, **36**(3–4), 221–242.

Examples

```
data(octane)

pca=PcaHubert(octane, k=10)
screeplot(pca, type="lines")

pca2 <- PcaHubert(octane, k=2)
plot(pca2, id.n.sd=6)

pca7 <- PcaHubert(octane, k=7)
plot(pca7, id.n.sd=6)
```

`olitos` *Olive Oil Data*

Description

This dataset consists of 120 olive oil samples on measurements on 25 chemical compositions (fatty acids, sterols, triterpenic alcohols) of olive oils from Tuscany, Italy (Armanino et al. 1989). There are 4 classes corresponding to different production areas. Class 1, Class 2, Class 3, and Class 4 contain 50, 25, 34, and 11 observations, respectively.

Usage

```
data(olitos)
```

Format

A data frame with 120 observations on the following 26 variables.

`X1` Free fatty acids

`X2` Refractive index

`X3` K268

`X4` delta K

- `X5` Palmitic acid
- `X6` Palmitoleic acid
- `X7` a numeric vector
- `X8` a numeric vector
- `X9` a numeric vector
- `X10` a numeric vector
- `X11` a numeric vector
- `X12` a numeric vector
- `X13` a numeric vector
- `X14` a numeric vector
- `X15` a numeric vector
- `X16` a numeric vector
- `X17` a numeric vector
- `X18` a numeric vector
- `X19` a numeric vector
- `X20` a numeric vector
- `X21` a numeric vector
- `X22` a numeric vector
- `X23` a numeric vector
- `X24` a numeric vector
- `X25` a numeric vector
- `grp` a factor with levels `1` `2` `3` `4`

Source

Prof. Roberto Todeschini, Milano Chemometrics and QSAR Research Group http://michem.disat.unimib.it/chm/download/datasets.htm

References

C. Armanino, R. Leardi, S. Lanteri and G. Modi, 1989. Chemometric analysis of Tuscan olive oils. *Cbemometrics and Intelligent Laboratoty Sysiem*, 5: 343–354.

R. Todeschini, V. Consonni, A. Mauri, M. Pavan (2004) Software for the calculation of molecular descriptors. Pavan M. Talete slr, Milan, Italy, http://www.talete.mi.it

Examples

```
data(olitos)
cc <- Linda(grp~., data=olitos, method="mcdC", l1med=TRUE)
cc
pr <- predict(cc)
tt <- mtxconfusion(cc@grp, pr@classification, printit=TRUE)
```

OsloTransect *Oslo Transect Data*

Description

The oslo Transect data set contains 360 samples of different plant species collected along a 120 km transect running through the city of Oslo, Norway.

Usage

```
data(OsloTransect)
```

Format

A data frame with 360 observations on the following 38 variables.

- `X.ID` a numeric vector, unique ID of the sample
- `X.MAT` a factor with levels `BBA` `BIL` `BWO` `FER` `MOS` `ROG` `SNE` `STW` `TWI`
- `XCOO` a numeric vector, X coordinate
- `YCOO` a numeric vector, Y coordinate
- `XCOO_km` a numeric vector
- `YCOO_km` a numeric vector
- `X.FOREST` a factor with levels `BIRSPR` `MIXDEC` `PINE` `SPRBIR` `SPRPIN` `SPRUCE`
- `DAY` a numeric vector
- `X.WEATHER` a factor with levels `CLOUD` `MOIST` `NICE` `RAIN`
- `ALT` a numeric vector
- `X.ASP` a factor with levels ` E` `FLAT` `N` `NE` `NW` `S` `SE` `SW` `W`
- `X.GRVEG` a factor with levels `BLGR` `BLLY` `BLMOLI` `BLUE` `BLUGRA` `GRAS` `GRBLU` `GRFE` `GRMO` `LYLI` `MIX` `MOGR` `MOSS`
- `X.FLITHO` a factor with levels `CAMSED` `GNEID_O` `GNEIS_O` `GNEIS_R` `MAGM` `MICSH`
- `Ag_ppb` a numeric vector
- `As_ash` a numeric vector
- `B` a numeric vector
- `Ba` a numeric vector
- `Ca` a numeric vector
- `Cd` a numeric vector
- `Co` a numeric vector
- `Cr` a numeric vector
- `Cu` a numeric vector
- `Fe` a numeric vector

`Hg_ppb` a numeric vector

`K` a numeric vector

`La` a numeric vector

`LOI` a numeric vector

`Mg` a numeric vector

`Mn` a numeric vector

`Mo` a numeric vector

`Ni` a numeric vector

`P` a numeric vector

`Pb` a numeric vector

`S` a numeric vector

`Sb` a numeric vector

`Sr` a numeric vector

`Ti` a numeric vector

`Zn` a numeric vector

Details

Samples of different plant species were collected along a 120 km transect running through the city of Oslo, Norway (forty samples each of leaves, needles,roots or barks of several plant species), and the concentrations of 25 chemical elements for the sample materials are reported. The factors that influenced the observed element concentrations in the sample materials were investigated. This data set was used in Todorov and Filzmoser (2007) for illustration of the robust statistics for one-way MANOVA implemented in the function `Wilks.test`.

Source

REIMANN,C., ARNOLDUSSEN,A., BOYD,R., FINNE,T.E., NORDGULEN,Oe., VOLDEN,T. and ENGLMAIER,P. (2006) The Influence of a city on element contents of a terrestrial moss (Hylocomium splendens), *The Science of the Total Environment* **369** 419–432.

REIMANN,C., ARNOLDUSSEN,A., BOYD,R., FINNE,T.E., KOLLER,F., NORDGULEN,Oe., and ENGLMAIER,P. (2007) Element contents in leaves of four plant species (birch, mountain ash, fern and spruce) along anthropogenic and geogenic concentration gradients, *The Science of the Total Environment* **377** 416–433.

REIMANN,C., ARNOLDUSSEN,A., FINNE,T.E., KOLLER,F., NORDGULEN,Oe., and ENGLMAIER,P., (2007) Element contents in birch leaves, bark and wood under different anthropogenic and geogenic conditions, *Applied Geochemistry*, **22** 1549–1566.

References

Todorov V. and Filzmoser P. (2007) Robust statistic for the one-way MANOVA, *submetted to the Journal of Environmetrics*.

Examples

```
data(OsloTransect)
str(OsloTransect)

##
##  Log-transform the numerical part of the data,
##  choose the desired groups and variables and
##  perform the classical Wilks' Lambda test
##
OsloTransect[,14:38] <- log(OsloTransect[,14:38])
grp <- OsloTransect$X.FLITHO
ind <- which(grp =="CAMSED" | grp == "GNEIS_O" |
    grp == "GNEIS_R" | grp=="MAGM")
(cwl <- Wilks.test(X.FLITHO~K+P+Zn+Cu,data=OsloTransect[ind,]))

##
## Perform now the robust MCD based Wilks' Lambda test.
##  Use the already computed multiplication factor 'xd' and
##  degrees of freedom 'xq' for the approximate distribution.
##

xd <- -0.003708238
xq <- 11.79073
(mcdwl <- Wilks.test(X.FLITHO~K+P+Zn+Cu,data=OsloTransect[ind,],
    method="mcd", xd=xd, xq=xq))
```

Pca-class *Class "Pca" - virtual base class for all classic and robust PCA classes*

Description

The class Pca searves as a base class for deriving all other classes representing the results of the classical and robust Principal Component Analisys methods

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

call: Object of class "language"

center: Object of class "vector" the center of the data

scale: Object of class "vector" the scaling applied to each variable of the data

loadings: Object of class "matrix" the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors)

eigenvalues: Object of class "vector" the eigenvalues `scores`**:** Object of class `"matrix"` the scores - the value of the projected on the space of the principal components data (the centred (and scaled if requested) data multiplied by the `loadings` matrix) is returned. Hence, `cov(scores)` is the diagonal matrix `diag(eigenvalues)`

`k`**:** Object of class `"numeric"` number of (choosen) principal components

`sd`**:** Object of class `"Uvector"` Score distances within the robust PCA subspace

`od`**:** Object of class `"Uvector"` Orthogonal distances to the robust PCA subspace

`cutoff.sd`**:** Object of class `"numeric"` Cutoff value for the score distances

`cutoff.od`**:** Object of class `"numeric"` Cutoff values for the orthogonal distances

`flag`**:** Object of class `"Uvector"` The observations whose score distance is larger than cutoff.sd or whose orthogonal distance is larger than cutoff.od can be considered as outliers and receive a flag equal to zero. The regular observations receive a flag 1

crit.pca.distances criterion to use for computing the cutoff values for the orthogonal and score distances. Default is 0.975.

`n.obs`**:** Object of class `"numeric"` the number of observations

Methods

getCenter `signature(obj = "Pca")`: center of the data

getScale `signature(obj = "Pca")`: return the scaling applied to each variable

getEigenvalues `signature(obj = "Pca")`: the eigenvalues of the covariance/correlation matrix, though the calculation is actually done with the singular values of the data matrix)

getLoadings `signature(obj = "Pca")`: returns the matrix `loadings` (i.e., a matrix whose columns contain the eigenvectors). The function prcomp returns this matrix in the element rotation.

getPrcomp `signature(obj = "Pca")`: returns an S3 object `prcomp` for compatibility with the functions prcomp() and princomp(). Thus the standard plots screeplot() and biplot() can be used

getScores `signature(obj = "Pca")`: returns the rotated data (the centred (and scaled if requested) data multiplied by the loadings matrix).

getSdev `signature(obj = "Pca")`: returns the standard deviations of the principal components (i.e., the square roots of the eigenvalues of the covariance/correlation matrix, though the calculation is actually done with the singular values of the data matrix)

plot `signature(x = "Pca")`: produces a distance plot (if `k=rank`) or distance-distance plot (if`k<rank`)

print `signature(x = "Pca")`: prints the results. The difference to the show() method is that additional parametesr are possible.

show `signature(object = "Pca")`: prints the results

predict `signature(object = "Pca")`: calculates prediction using the results in `object`. An optional data frame or matrix in which to look for variables with which to predict. If omitted, the scores are used. If the original fit used a formula or a data frame or a matrix with column names, newdata must contain columns with the same names. Otherwise it must contain the same number of columns, to be used in the same order. See also `predict.prcomp` and `predict.princomp`

screeplot `signature(x = "Pca")`: plots the variances against the number of the principal component. See also `plot.prcomp` and `plot.princomp`

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

PcaClassic, PcaClassic-class, PcaRobust-class

Examples

```
showClass("Pca")
```

---

pca.distances — *Compute score and orthogonal distances for Principal Components (objects of class 'Pca')*

Description

Compute score and orthogonal distances for an object (derived from)Pca-class.

Usage

```
pca.distances(obj, data, r, crit=0.975)
```

Arguments

obj	an object of class (derived from) "Pca".
data	The data matrix for which the "Pca" object was computed.
r	rank of data
crit	Criterion to use for computing the cutoff values.

Details

This function calculates the score and orthogonal distances and the appropriate cutoff values for identifying outlying observations. The computed values are used to create a vector a of flags, one for each observation, identifying the outliers.

Value

An S4 object of class derived from the virtual class Pca-class - the same object passed to the function, but with the score and orthogonal distances as well as their cutoff values and the corresponding flags appended to it.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

M. Hubert, P. J. Rousseeuw, K. Vanden Branden (2005), ROBPCA: a new approach to robust principal components analysis, *Technometrics*, **47**, 64–79.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## PCA of the Hawkins Bradu Kass's Artificial Data
##  using all 4 variables
data(hbk)
pca <- PcaHubert(hbk)
pca.distances(pca, hbk, rankMM(hbk))
```

---

pca.scoreplot — *Score plot for Principal Components (objects of class 'Pca')*

Description

Produces a score plot from an object (derived from) Pca-class.

Usage

```
pca.scoreplot(obj, i=1, j=2, main, id.n=0, ...)
```

Arguments

obj	an object of class (derived from) "Pca".
i	First score coordinate, defaults to i=1.
j	Second score coordinate, defaults to j=2.
main	The main title of the plot.
id.n	Number of observations to identify by a label. Defaults to id.n=0.
...	optional arguments to be passed to the internal graphical functions.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

See Also

Pca-class, PcaClassic, PcaRobust-class.

Examples

```
require(graphics)

## PCA of the Hawkins Bradu Kass's Artificial Data
##  using all 4 variables
data(hbk)
pca <- PcaHubert(hbk)
pca
pca.scoreplot(pca)
```

PcaClassic *Principal Components Analysis*

Description

Performs a principal components analysis and returns the results as an object of class PcaClassic (aka constructor).

Usage

```
PcaClassic(x, ...)
## Default S3 method:
PcaClassic(x, k = ncol(x), kmax = ncol(x),
    scale=FALSE, signflip=TRUE, crit.pca.distances = 0.975, trace=FALSE, ...)
## S3 method for class 'formula'
PcaClassic(formula, data = NULL, subset, na.action, ...)
```

Arguments

`formula`	a formula with no response variable, referring only to numeric variables.
`data`	an optional data frame (or similar: see `model.frame`) containing the variables in the formula `formula`.
`subset`	an optional vector used to select rows (observations) of the data matrix `x`.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The default is `na.omit`.
`...`	arguments passed to or from other methods.
`x`	a numeric matrix (or data frame) which provides the data for the principal components analysis.
`k`	number of principal components to compute. If `k` is missing, or `k = 0`, the algorithm itself will determine the number of components by finding such `k` that $l_k/l_1 >= 10.E-3$ and $\Sigma_{j=1}^k l_j/\Sigma_{j=1}^r l_j >= 0.8$. It is preferable to investigate the scree plot in order to choose the number of components and then run again. Default is `k=ncol(x)`.
`kmax`	maximal number of principal components to compute. Default is `kmax=10`. If `k` is provided, `kmax` does not need to be specified, unless `k` is larger than 10.

scale
: a value indicating whether and how the variables should be scaled to have unit variance (only possible if there are no constant variables). If `scale=FALSE` (default) or `scale=NULL` no scaling is performed (a vector of 1s is returned in the scale slot). If `scale=TRUE` the data are scaled to have unit variance. Alternatively it can be a function like `sd` or `Qn` or a vector of length equal the number of columns of `x`. The value is passed to the underlying function and the result returned is stored in the scale slot. Default is `scale=FALSE`.

signflip
: a logical value indicating wheather to try to solve the sign indeterminancy of the loadings - ad hoc approach setting the maximum element in a singular vector to be positive. Default is `signflip = FALSE`

crit.pca.distances
: criterion to use for computing the cutoff values for the orthogonal and score distances. Default is 0.975.

trace
: whether to print intermediate results. Default is `trace = FALSE`

Value

An S4 object of class `PcaClassic-class` which is a subclass of the virtual class `Pca-class`.

Note

This function can be seen as a wrapper arround prcomp() from `stats` which returns the results of the PCA in a class compatible with the object model for robust PCA.

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

See Also

`Pca-class`, `PcaClassic-class`,

PcaClassic-class *Class "PcaClassic" - Principal Components Analysis*

Description

Contains the results of a classical Principal Components Analysis

Objects from the Class

Objects can be created by calls of the form `new("PcaClassic",...)` but the usual way of creating `PcaClassic` objects is a call to the function `PcaClassic` which serves as a constructor.

Slots

- `call`: Object of class "language"
- `center`: Object of class "vector" the center of the data
- `scale`: Object of class "vector" the scaling applied to each variable
- `loadings`: Object of class "matrix" the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors)
- `eigenvalues`: Object of class "vector" the eigenvalues
- `scores`: Object of class "matrix" the scores - the value of the projected on the space of the principal components data (the centred (and scaled if requested) data multiplied by the `loadings` matrix) is returned. Hence, `cov(scores)` is the diagonal matrix `diag(eigenvalues)`
- `k`: Object of class "numeric" number of (choosen) principal components
- `sd`: Object of class "Uvector" Score distances within the robust PCA subspace
- `od`: Object of class "Uvector" Orthogonal distances to the robust PCA subspace
- `cutoff.sd`: Object of class "numeric" Cutoff value for the score distances
- `cutoff.od`: Object of class "numeric" Cutoff values for the orthogonal distances
- `flag`: Object of class "Uvector" The observations whose score distance is larger than cutoff.sd or whose orthogonal distance is larger than cutoff.od can be considered as outliers and receive a flag equal to zero. The regular observations receive a flag 1
- `n.obs`: Object of class "numeric" the number of observations

Extends

Class "Pca", directly.

Methods

getQuan `signature(obj = "PcaClassic")`: returns the number of observations used in the computation, i.e. n.obs

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

PcaRobust-class, Pca-class, PcaClassic

Examples

```
showClass("PcaClassic")
```

PcaCov *Robust PCA based on a robust covariance matrix*

Description

Robust PCA are obtained by replacing the classical covariance matrix by a robust covariance estimator. This can be one of the available in rrcov estimators, i.e. MCD, OGK, M or S estimator.

Usage

```
PcaCov(x, ...)
## Default S3 method:
PcaCov(x, k = ncol(x), kmax = ncol(x), cov.control=CovControlMcd(),
    scale = FALSE, signflip = TRUE, crit.pca.distances = 0.975, trace=FALSE, ...)
## S3 method for class 'formula'
PcaCov(formula, data = NULL, subset, na.action, ...)
```

Arguments

formula	a formula with no response variable, referring only to numeric variables.
data	an optional data frame (or similar: see model.frame) containing the variables in the formula formula.
subset	an optional vector used to select rows (observations) of the data matrix x.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The default is na.omit.
...	arguments passed to or from other methods.
x	a numeric matrix (or data frame) which provides the data for the principal components analysis.
k	number of principal components to compute. If k is missing, or k = 0, the algorithm itself will determine the number of components by finding such k that $l_k/l_1 >= 10.E-3$ and $\Sigma_{j=1}^k l_j/\Sigma_{j=1}^r l_j >= 0.8$. It is preferable to investigate the scree plot in order to choose the number of components and then run again. Default is k=ncol(x).
kmax	maximal number of principal components to compute. Default is kmax=10. If k is provided, kmax does not need to be specified, unless k is larger than 10.
cov.control	specifies which covariance estimator to use by providing a CovControl-class object. The default is CovControlMcd-class which will indirectly call CovMcd. If cov.control=NULL is specified, the classical estimates will be used by calling CovClassic.
scale	a value indicating whether and how the variables should be scaled to have unit variance (only possible if there are no constant variables). If scale=FALSE (default) or scale=NULL no scaling is performed (a vector of 1s is returned in the scale slot). If scale=TRUE the data are scaled by the estimator used to compute

the covariance matrix (MCD by default). Alternatively it can be a function like sd or Qn or a vector of length equal the number of columns of x. The value is passed to the underlying function and the result returned is stored in the scale slot. Default is scale=FALSE.

signflip a logical value indicating wheather to try to solve the sign indeterminancy of the loadings - ad hoc approach setting the maximum element in a singular vector to be positive. Default is signflip = FALSE

crit.pca.distances criterion to use for computing the cutoff values for the orthogonal and score distances. Default is 0.975.

trace whether to print intermediate results. Default is trace = FALSE

Details

PcaCov, serving as a constructor for objects of class PcaCov-class is a generic function with "formula" and "default" methods. For details see the relevant references.

Value

An S4 object of class PcaCov-class which is a subclass of the virtual class PcaRobust-class.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## PCA of the Hawkins Bradu Kass's Artificial Data
##  using all 4 variables
    data(hbk)
    pca <- PcaCov(hbk)
    pca

## Compare with the classical PCA
    prcomp(hbk)

## or
    PcaClassic(hbk)

## If you want to print the scores too, use
    print(pca, print.x=TRUE)

## Using the formula interface
    PcaCov(~., data=hbk)
```

```
## To plot the results:

    plot(pca)                    # distance plot
    pca2 <- PcaCov(hbk, k=2)
    plot(pca2)                   # PCA diagnostic plot (or outlier map)

## Use the standard plots available for for prcomp and princomp
    screeplot(pca)
    biplot(pca)
```

PcaCov-class *Class "PcaCov" - Robust PCA based on a robust covariance matrix*

Description

Robust PCA are obtained by replacing the classical covariance matrix by a robust covariance estimator. This can be one of the available in `rrcov` estimators, i.e. MCD, OGK, M, S or Stahel-Donoho estimator.

Objects from the Class

Objects can be created by calls of the form `new("PcaCov",...)` but the usual way of creating `PcaCov` objects is a call to the function `PcaCov` which serves as a constructor.

Slots

`quan`: Object of class `"numeric"` The quantile `h` used throughout the algorithm

`call`, `center`, `loadings`, `eigenvalues`, `scores`, `k`, `sd`, `od`, `cutoff.sd`, `cutoff.od`, `flag`, `n.obs`: from the "`Pca`" class.

Extends

Class "`PcaRobust`", directly. Class "`Pca`", by class "PcaRobust", distance 2.

Methods

getQuan `signature(obj = "PcaCov")`: ...

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

PcaRobust-class, Pca-class, PcaClassic, PcaClassic-class

Examples

```
showClass("PcaCov")
```

PcaGrid *Robust Principal Components based on Projection Pursuit (PP): GRID search Algorithm*

Description

Computes an approximation of the PP-estimators for PCA using the grid search algorithm in the plane.

Usage

```
    PcaGrid(x, ...)
    ## Default S3 method:
PcaGrid(x, k = 0, kmax = ncol(x),
     scale=FALSE, na.action = na.fail, crit.pca.distances = 0.975, trace=FALSE, ...)
    ## S3 method for class 'formula'
PcaGrid(formula, data = NULL, subset, na.action, ...)
```

Arguments

formula	a formula with no response variable, referring only to numeric variables.
data	an optional data frame (or similar: see model.frame) containing the variables in the formula formula.
subset	an optional vector used to select rows (observations) of the data matrix x.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The default is na.omit.
...	arguments passed to or from other methods.
x	a numeric matrix (or data frame) which provides the data for the principal components analysis.
k	number of principal components to compute. If k is missing, or k = 0, it is set to the number of columns of the data. It is preferable to investigate the scree plot in order to choose the number of components and then run again. Default is k=0.
kmax	maximal number of principal components to compute. Default is kmax=10. If k is provided, kmax does not need to be specified, unless k is larger than 10.

scale a value indicating whether and how the variables should be scaled. If scale = FALSE (default) or scale = NULL no scaling is performed (a vector of 1s is returned in the scale slot). If scale = TRUE the data are scaled to have unit variance. Alternatively it can be a function like sd or mad or a vector of length equal the number of columns of x. The value is passed to the underlying function and the result returned is stored in the scale slot. Default is scale = FALSE

crit.pca.distances
criterion to use for computing the cutoff values for the orthogonal and score distances. Default is 0.975.

trace whether to print intermediate results. Default is trace = FALSE

Details

PcaGrid, serving as a constructor for objects of class PcaGrid-class is a generic function with "formula" and "default" methods. For details see PCAgrid and the relevant references.

Value

An S4 object of class PcaGrid-class which is a subclass of the virtual class PcaRobust-class.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

C. Croux, P. Filzmoser, M. Oliveira, (2007). Algorithms for Projection-Pursuit Robust Principal Component Analysis, *Chemometrics and Intelligent Laboratory Systems*, 87, 225.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
# multivariate data with outliers
library(mvtnorm)
x <- rbind(rmvnorm(200, rep(0, 6), diag(c(5, rep(1,5)))),
           rmvnorm( 15, c(0, rep(20, 5)), diag(rep(1, 6))))
# Here we calculate the principal components with PCAgrid
pc <- PcaGrid(x, 6)
# we could draw a biplot too:
biplot(pc)

# we could use another objective function, and
# maybe only calculate the first three principal components:
pc <- PcaGrid(x, 3, method="qn")
biplot(pc)

# now we want to compare the results with the non-robust principal components
pc <- PcaClassic(x, k=3)
# again, a biplot for comparision:
biplot(pc)
```

PcaGrid-class *Class "PcaGrid" - Robust PCA using PP - GRID search Algorithm*

Description

Holds the results of an approximation of the PP-estimators for PCA using the grid search algorithm in the plane.

Objects from the Class

Objects can be created by calls of the form `new("PcaGrid",...)` but the usual way of creating `PcaGrid` objects is a call to the function `PcaGrid()` which serves as a constructor.

Slots

`call`, `center`, `scale`, `loadings`, `eigenvalues`, `scores`, `k`, `sd`, `od`, `cutoff.sd`, `cutoff.od`, `flag`, `n.obs`: from the "`Pca`" class.

Extends

Class "`PcaRobust`", directly. Class "`Pca`", by class "`PcaRobust`", distance 2.

Methods

getQuan `signature(obj = "PcaGrid")`: ...

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

See Also

`PcaRobust-class`, `Pca-class`, `PcaClassic`, `PcaClassic-class`

Examples

```
showClass("PcaGrid")
```

PcaHubert *ROBPCA - ROBust method for Principal Components Analysis*

Description

The ROBPCA algorithm was proposed by Hubert et al (2005) and stays for 'ROBust method for Principal Components Analysis'. It is resistant to outliers in the data. The robust loadings are computed using projection-pursuit techniques and the MCD method. Therefore ROBPCA can be applied to both low and high-dimensional data sets. In low dimensions, the MCD method is applied.

Usage

```
PcaHubert(x, ...)
## Default S3 method:
PcaHubert(x, k = 0, kmax = 10, alpha = 0.75, mcd = TRUE,
maxdir=250, scale = FALSE, signflip = TRUE, crit.pca.distances = 0.975, trace=FALSE, ...)
## S3 method for class 'formula'
PcaHubert(formula, data = NULL, subset, na.action, ...)
```

Arguments

`formula`	a formula with no response variable, referring only to numeric variables.
`data`	an optional data frame (or similar: see `model.frame`) containing the variables in the formula `formula`.
`subset`	an optional vector used to select rows (observations) of the data matrix `x`.
`na.action`	a function which indicates what should happen when the data contain `NA`s. The default is set by the `na.action` setting of `options`, and is `na.fail` if that is unset. The default is `na.omit`.
`...`	arguments passed to or from other methods.
`x`	a numeric matrix (or data frame) which provides the data for the principal components analysis.
`k`	number of principal components to compute. If `k` is missing, or `k = 0`, the algorithm itself will determine the number of components by finding such `k` that $l_k/l_1 >= 10.E-3$ and $\Sigma_{j=1}^k l_j/\Sigma_{j=1}^r l_j >= 0.8$. It is preferable to investigate the scree plot in order to choose the number of components and then run again. Default is `k=0`.
`kmax`	maximal number of principal components to compute. Default is `kmax=10`. If `k` is provided, `kmax` does not need to be specified, unless `k` is larger than 10.
`alpha`	this parameter measures the fraction of outliers the algorithm should resist. In MCD alpha controls the size of the subsets over which the determinant is minimized, i.e. alpha*n observations are used for computing the determinant. Allowed values are between 0.5 and 1 and the default is 0.75.

mcd
: Logical - when the number of variables is sufficiently small, the loadings are computed as the eigenvectors of the MCD covariance matrix, hence the function CovMcd() is automatically called. The number of principal components is then taken as k = rank(x). Default is mcd=TRUE. If mcd=FALSE, the ROBPCA algorithm is always applied.

maxdir
: maximal number of random directions to use for computing the outlyingness of the data points. Default is maxdir=250. If the number n of observations is small all possible n*(n-1)/2 pairs of observations are taken to generate the directions.

scale
: a value indicating whether and how the variables should be scaled. If scale=FALSE (default) or scale=NULL no scaling is performed (a vector of 1s is returned in the scale slot). If scale=TRUE the data are scaled to have unit variance. Alternatively it can be a function like sd or mad or a vector of length equal the number of columns of x. The value is passed to the underlying function and the result returned is stored in the scale slot. Default is scale=FALSE.

signflip
: a logical value indicating wheather to try to solve the sign indeterminancy of the loadings - ad hoc approach setting the maximum element in a singular vector to be positive. Default is signflip = FALSE

crit.pca.distances
: criterion to use for computing the cutoff values for the orthogonal and score distances. Default is 0.975.

trace
: whether to print intermediate results. Default is trace = FALSE

Details

PcaHubert, serving as a constructor for objects of class PcaHubert-class is a generic function with "formula" and "default" methods. The calculation is done using the ROBPCA method of Hubert et al (2005) which can be described briefly as follows. For details see the relevant references.

Let n denote the number of observations, and p the number of original variables in the input data matrix X. The ROBPCA algorithm finds a robust center M (p x 1) of the data and a loading matrix P which is (p x k) dimensional. Its columns are orthogonal and define a new coordinate system. The scores T, an (n x k) matrix, are the coordinates of the centered observations with respect to the loadings:

$$T = (X - M)P$$

The ROBPCA algorithm also yields a robust covariance matrix (often singular) which can be computed as

$$S = PLP^t$$

where L is the diagonal matrix with the eigenvalues $l_1, \dots, _k$.

This is done in the following three main steps:

Step 1: The data are preprocessed by reducing their data space to the subspace spanned by the n observations. This is done by singular value decomposition of the input data matrix. As a result the data are represented using at most n-1=rank(X) without loss of information.

Step 2: In this step for each data point a measure of outlyingness is computed. For this purpose the high-dimensional data points are projected on many univariate directions, each time the univariate

MCD estimator of location and scale is computed and the standardized distance to the center is measured. The largest of these distances (over all considered directions) is the outlyingness measure of the data point. The h data points with smallest outlyingness measure are used to compute the covariance matrix Σ_h and to select the number k of principal components to retain. This is done by finding such k that $l_k/l_1 >= 10.E-3$ and $\Sigma_{j=1}^{k} l_j/\Sigma_{j=1}^{r} l_j >= 0.8$ Alternatively the number of principal components k can be specified by the user after inspecting the scree plot.

Step 3: The data points are projected on the k-dimensional subspace spanned by the k eigenvectors corresponding to the largest k eigenvalues of the matrix Σ_h. The location and scatter of the projected data are computed using the reweighted MCD estimator and the eigenvectors of this scatter matrix yield the robust principal components.

Value

An S4 object of class `PcaHubert-class` which is a subclass of the virtual class `PcaRobust-class`.

Note

The ROBPCA algorithm is implemented on the bases of the Matlab implementation, available as part of *LIBRA, a Matlab Library for Robust Analysis* to be found at www.wis.kuleuven.ac.be/stat/robust.html

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

M. Hubert, P. J. Rousseeuw, K. Vanden Branden (2005), ROBPCA: a new approach to robust principal components analysis, *Technometrics*, **47**, 64–79.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## PCA of the Hawkins Bradu Kass's Artificial Data
##  using all 4 variables
    data(hbk)
    pca <- PcaHubert(hbk)
    pca

## Compare with the classical PCA
    prcomp(hbk)

## or
    PcaClassic(hbk)

## If you want to print the scores too, use
    print(pca, print.x=TRUE)

## Using the formula interface
    PcaHubert(~., data=hbk)
```

```
## To plot the results:

    plot(pca)                    # distance plot
    pca2 <- PcaHubert(hbk, k=2)
    plot(pca2)                   # PCA diagnostic plot (or outlier map)

## Use the standard plots available for prcomp and princomp
    screeplot(pca)
    biplot(pca)

## Restore the covraiance matrix
    py <- PcaHubert(hbk)
    cov.1 <- py@loadings %*% diag(py@eigenvalues) %*% t(py@loadings)
    cov.1
```

`PcaHubert-class` *Class "PcaHubert" - ROBust method for Principal Components Analysis*

Description

The ROBPCA algorithm was proposed by Hubert et al (2005) and stays for 'ROBust method for Principal Components Analysis'. It is resistant to outliers in the data. The robust loadings are computed using projection-pursuit techniques and the MCD method. Therefore ROBPCA can be applied to both low and high-dimensional data sets. In low dimensions, the MCD method is applied.

Objects from the Class

Objects can be created by calls of the form `new("PcaHubert",...)` but the usual way of creating `PcaHubert` objects is a call to the function `PcaHubert` which serves as a constructor.

Slots

`alpha`**:** Object of class `"numeric"` the fraction of outliers the algorithm should resist - this is the argument alpha

`quan`**:** Object of class `"numeric"` The quantile `h` used throughout the algorithm

`call`, `center`, `loadings`, `eigenvalues`, `scores`, `k`, `sd`, `od`, `cutoff.sd`, `cutoff.od`, `flag`, `n.obs`**:** from the `"Pca"` class.

Extends

Class `"PcaRobust"`, directly. Class `"Pca"`, by class "PcaRobust", distance 2.

Methods

getQuan `signature(obj = "PcaHubert")`: Returns the quantile used throughout the algorithm

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

PcaRobust-class, Pca-class, PcaClassic, PcaClassic-class

Examples

```
showClass("PcaHubert")
```

PcaLocantore *Spherical Principal Components*

Description

The Spherical Principal Components procedure was proposed by Locantore et al., (1999) as a functional data analysis method. The idea is to perform classical PCA on the data, \ projected onto a unit sphere. The estimates of the eigenvectors are consistent and the procedure is extremely fast. The simulations of Maronna (2005) show that this method has very good performance.

Usage

```
PcaLocantore(x, ...)
## Default S3 method:
PcaLocantore(x, k = ncol(x), kmax = ncol(x), delta = 0.001,
    na.action = na.fail, scale = FALSE, signflip = TRUE,
    crit.pca.distances = 0.975, trace=FALSE, ...)
## S3 method for class 'formula'
PcaLocantore(formula, data = NULL, subset, na.action, ...)
```

Arguments

formula	a formula with no response variable, referring only to numeric variables.
data	an optional data frame (or similar: see model.frame) containing the variables in the formula formula.
subset	an optional vector used to select rows (observations) of the data matrix x.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The default is na.omit.
...	arguments passed to or from other methods.

x	a numeric matrix (or data frame) which provides the data for the principal components analysis.
k	number of principal components to compute. If k is missing, or k = 0, the algorithm itself will determine the number of components by finding such k that $l_k/l_1 >= 10.E-3$ and $\Sigma_{j=1}^k l_j/\Sigma_{j=1}^r l_j >= 0.8$. It is preferable to investigate the scree plot in order to choose the number of components and then run again. Default is k=ncol(x).
kmax	maximal number of principal components to compute. Default is kmax=10. If k is provided, kmax does not need to be specified, unless k is larger than 10.
delta	an accuracy parameter
scale	a value indicating whether and how the variables should be scaled to have unit variance (only possible if there are no constant variables). If scale=FALSE (default) or scale=NULL no scaling is performed (a vector of 1s is returned in the scale slot). If scale=TRUE the data are scaled by mad. Alternatively it can be a function like sd or Qn or a vector of length equal the number of columns of x. The value is passed to the underlying function and the result returned is stored in the scale slot. Default is scale=FALSE.
signflip	a logical value indicating wheather to try to solve the sign indeterminancy of the loadings - ad hoc approach setting the maximum element in a singular vector to be positive. Default is signflip = FALSE
crit.pca.distances	criterion to use for computing the cutoff values for the orthogonal and score distances. Default is 0.975.
trace	whether to print intermediate results. Default is trace = FALSE

Details

PcaLocantore, serving as a constructor for objects of class PcaLocantore-class is a generic function with "formula" and "default" methods. For details see the relevant references.

Value

An S4 object of class PcaLocantore-class which is a subclass of the virtual class PcaRobust-class.

Author(s)

Valentin Todorov <valentin.todorov@chello.at> The SPC algorithm is implemented on the bases of the available from the web site of the book Maronna et al. (2006) code http://www.wiley.com/legacy/wileychi/robust_statistics/

References

N. Locantore, J. Marron, D. Simpson, N. Tripoli, J. Zhang and K. Cohen K. (1999), Robust principal components for functional data. Test, 8, 1-28.

R. Maronna, D. Martin and V. Yohai (2006), Robust Statistics: Theory and Methods. Wiley, New York.

R. Maronna (2005). Principal components and orthogonal regression based on robust scales. Technometrics, 47, 264-273.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
## PCA of the Hawkins Bradu Kass's Artificial Data
##  using all 4 variables
    data(hbk)
    pca <- PcaLocantore(hbk)
    pca

## Compare with the classical PCA
    prcomp(hbk)

## or
    PcaClassic(hbk)

## If you want to print the scores too, use
    print(pca, print.x=TRUE)

## Using the formula interface
    PcaLocantore(~., data=hbk)

## To plot the results:

    plot(pca)                    # distance plot
    pca2 <- PcaLocantore(hbk, k=2)
    plot(pca2)                   # PCA diagnostic plot (or outlier map)

## Use the standard plots available for for prcomp and princomp
    screeplot(pca)
    biplot(pca)
```

PcaLocantore-class   *Class "PcaLocantore" Spherical Principal Components*

Description

The Spherical Principal Components procedure was proposed by Locantore et al., (1999) as a functional data analysis method. The idea is to perform classical PCA on the the data, \ projected onto a unit sphere. The estimates of the eigenvectors are consistent and the procedure is extremly fast. The simulations of Maronna (2005) show that this method has very good performance.

Objects from the Class

Objects can be created by calls of the form `new("PcaLocantore",...)` but the usual way of creating `PcaLocantore` objects is a call to the function `PcaLocantore` which serves as a constructor.

Slots

delta: Accuracy parameter

quan: Object of class "numeric" The quantile h used throughout the algorithm

call, center, scale, loadings, eigenvalues, scores, k, sd, od, cutoff.sd, cutoff.od, flag, n.obs: from the "Pca" class.

Extends

Class "PcaRobust", directly. Class "Pca", by class "PcaRobust", distance 2.

Methods

getQuan `signature(obj = "PcaLocantore")`: ...

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

PcaRobust-class, Pca-class, PcaClassic, PcaClassic-class

Examples

```
showClass("PcaLocantore")
```

PcaProj — *Robust Principal Components based on Projection Pursuit (PP): Croux and Ruiz-Gazen (2005) algorithm*

Description

A fast and simple algorithm for approximating the PP-estimators for PCA: Croux and Ruiz-Gazen (2005)

Usage

```
    PcaProj(x, ...)
    ## Default S3 method:
PcaProj(x, k = 0, kmax = ncol(x), scale=FALSE,
       na.action = na.fail, crit.pca.distances = 0.975, trace=FALSE, ...)
    ## S3 method for class 'formula'
PcaProj(formula, data = NULL, subset, na.action, ...)
```

Arguments

formula	a formula with no response variable, referring only to numeric variables.
data	an optional data frame (or similar: see model.frame) containing the variables in the formula formula.
subset	an optional vector used to select rows (observations) of the data matrix x.
na.action	a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The default is na.omit.
...	arguments passed to or from other methods.
x	a numeric matrix (or data frame) which provides the data for the principal components analysis.
k	number of principal components to compute. If k is missing, or k = 0, it is set to the number of columns of the data. It is preferable to investigate the scree plot in order to choose the number of components and then run again. Default is k=0.
kmax	maximal number of principal components to compute. Default is kmax=10. If k is provided, kmax does not need to be specified, unless k is larger than 10.
scale	a value indicating whether and how the variables should be scaled. If scale = FALSE (default) or scale = NULL no scaling is performed (a vector of 1s is returned in the scale slot). If scale = TRUE the data are scaled to have unit variance. Alternatively it can be a function like sd or mad or a vector of length equal the number of columns of x. The value is passed to the underlying function and the result returned is stored in the scale slot. Default is scale = FALSE
crit.pca.distances	criterion to use for computing the cutoff values for the orthogonal and score distances. Default is 0.975.
trace	whether to print intermediate results. Default is trace = FALSE

Details

PcaProj, serving as a constructor for objects of class PcaProj-class is a generic function with "formula" and "default" methods. For details see PCAproj and the relevant references.

Value

An S4 object of class PcaProj-class which is a subclass of the virtual class PcaRobust-class.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

C. Croux, A. Ruiz-Gazen (2005). High breakdown estimators for principal components: The projection-pursuit approach revisited, *Journal of Multivariate Analysis*, 95, 206–226.

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

Examples

```
# multivariate data with outliers
library(mvtnorm)
x <- rbind(rmvnorm(200, rep(0, 6), diag(c(5, rep(1,5)))),
           rmvnorm( 15, c(0, rep(20, 5)), diag(rep(1, 6))))
# Here we calculate the principal components with PcaProj
pc <- PcaProj(x, 6)
# we could draw a biplot too:
biplot(pc)

# we could use another calculation method and another objective function, and
# maybe only calculate the first three principal components:
pc <- PcaProj(x, k=3, method="qn", CalcMethod="sphere")
biplot(pc)

# now we want to compare the results with the non-robust principal components
pc <- PcaClassic(x, k=3)
# again, a biplot for comparision:
biplot(pc)
```

PcaProj-class *Class "PcaProj" - Robust PCA using PP - Croux and Ruiz-Gazen (2005) algorithm*

Description

Holds the results of an approximation of the PP-estimators for PCA by a fast and simple algorithm: Croux and Ruiz-Gazen (2005) algorithm.

Objects from the Class

Objects can be created by calls of the form `new("PcaProj",...)` but the usual way of creating `PcaProj` objects is a call to the function `PcaProj()` which serves as a constructor.

Slots

`call`, `center`, `scale`, `loadings`, `eigenvalues`, `scores`, `k`, `sd`, `od`, `cutoff.sd`, `cutoff.od`, `flag`, `n.obs`: from the "`Pca`" class.

Extends

Class "`PcaRobust`", directly. Class "`Pca`", by class "`PcaRobust`", distance 2.

Methods

getQuan `signature(obj = "PcaProj")`: ...

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

PcaRobust-class, Pca-class, PcaClassic, PcaClassic-class

Examples

```
showClass("PcaProj")
```

PcaRobust-class *Class "PcaRobust" is a virtual base class for all robust PCA classes*

Description

The class `PcaRobust` searves as a base class for deriving all other classes representing the results of the robust Principal Component Analisys methods

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`call`: Object of class `"language"`

`center`: Object of class `"vector"` the center of the data

`loadings`: Object of class `"matrix"` the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors)

`eigenvalues`: Object of class `"vector"` the eigenvalues

`scores`: Object of class `"matrix"` the scores - the value of the projected on the space of the principal components data (the centred (and scaled if requested) data multiplied by the `loadings` matrix) is returned. Hence, `cov(scores)` is the diagonal matrix `diag(eigenvalues)`

`k`: Object of class `"numeric"` number of (choosen) principal components

`sd`: Object of class `"Uvector"` Score distances within the robust PCA subspace

`od`: Object of class `"Uvector"` Orthogonal distances to the robust PCA subspace

`cutoff.sd`: Object of class `"numeric"` Cutoff value for the score distances

`cutoff.od`: Object of class `"numeric"` Cutoff values for the orthogonal distances

`flag`: Object of class `"Uvector"` The observations whose score distance is larger than cutoff.sd or whose orthogonal distance is larger than cutoff.od can be considered as outliers and receive a flag equal to zero. The regular observations receive a flag 1

`n.obs`: Object of class `"numeric"` the number of observations

Extends

Class "Pca", directly.

Methods

No methods defined with class "PcaRobust" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

Pca-class, PcaClassic-class,

Examples

```
showClass("PcaRobust")
```

---

plot-methods *Methods for Function 'plot' in Package 'rrcov'*

Description

Shows the Mahalanobis distances based on robust and/or classical estimates of the location and the covariance matrix in different plots. The following plots are available:

- index plot of the robust and mahalanobis distances

- distance-distance plot

- Chisquare QQ-plot of the robust and mahalanobis distances

- plot of the tolerance ellipses (robust and classic)

- Scree plot - Eigenvalues comparison plot

Usage

```
## S4 method for signature 'CovClassic'
plot(x, which = c("all","distance","qqchi2","tolellipse","screeplot"),
        ask=(which=="all" && dev.interactive()),
        cutoff, id.n, tol=1e-7, ...)
## S4 method for signature 'CovRobust'
plot(x, which = c("all","dd","distance","qqchi2","tolellipse","screeplot"),
        classic=FALSE, ask=(which=="all" && dev.interactive()),
        cutoff, id.n, tol=1e-7, ...)
```

Arguments

`x`	an object of class `"Cov"` or `"CovRobust"`
`which`	Which plot to show? See Details for description of the options. Default is `which="all"`. .
`classic`	whether to plot the classical distances too. Default is `classic=FALSE`. .
`ask`	logical; if 'TRUE', the user is *ask*ed before each plot, see 'par(ask=.)'. Default is `ask = which=="all" && dev.interactive()`.
`cutoff`	The cutoff value for the distances.
`id.n`	Number of observations to identify by a label. If not supplied, the number of observations with distance larger than `cutoff` is used.
`tol`	tolerance to be used for computing the inverse see 'solve'. Default is `tol = 10e-7`
`...`	other parameters to be passed through to plotting functions.

Methods

x = "Cov", y = "missing" Plot mahalanobis distances for `x`.

x = "CovRobust", y = "missing" Plot robust and classical mahalanobis distances for `x`.

Examples

```
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
cv <- CovClassic(hbk.x)
plot(cv)
rcv <- CovMest(hbk.x)
plot(rcv)
```

---

pottery *Archaic Greek Pottery data*

Description

The Archaic Greek Pottery data set contains data on fragments of Greek pottery which were classified into two groups according to their origin: Attic or Eritrean. Six chemical variables, metallic oxide constituents, were measured: Si, Al, Fe, Ca and Ti. The main data set consists of 13 Attic objects and 14 Eritrean ones. There is a separate data set with 13 observations which can be used as a test data set. It consists of 4 observations classified as "probably Attic" and the remaining 9 as "probably Eritrean".

Usage

```
data(pottery)
```

Format

Two data frames with 27 an 13 observations on the following 7 variables.

`SI` Si content

`AL` Al content

`FE` Fe content

`MG` Mg content

`CA` Ca content

`TI` Ti content

`origin` Origin - factor with two levels: `Attic` and `Eritrean`

Details

The Archaic Greek Pottery data set was first published by Stern and Descoeudres (1977) and later reproduced in Cooper and Weeks (1983) for illustration of linear discriminant analisys. The data set was used by Pires and Branco (2010) for illustration of their projection pursuit approach to linear discriminant analysis.

Source

STERN, W. B. and DESCOEUDRES, J.-P. (1977) X-RAY FLUORESCENCE ANALYSIS OF ARCHAIC GREEK POTTERY *Archaeometry*, Blackwell Publishing Ltd, **19**, 73–86.

References

Cooper, R.A. and Weekes, A.J.. 1983 *Data, Models, and Statistical Analysis*, (Lanham, MD: Rowman & Littlefield).

Pires, A. M. and A. Branco, J. (2010) Projection-pursuit approach to robust linear discriminant analysis *Journal Multivariate Analysis*, Academic Press, Inc., **101**, 2464–2485.

Examples

```
data(pottery)
x <- pottery[,c("MG", "CA")]
grp <- pottery$origin

##
## Compute robust location and covariance matrix and
## plot the tolerance ellipses
library(rrcov)
(mcd <- CovMcd(x))
col <- c(3,4)
gcol <- ifelse(grp == "Attic", col[1], col[2])
gpch <- ifelse(grp == "Attic", 16, 1)
plot(mcd, which="tolEllipsePlot", class=TRUE, col=gcol, pch=gpch)

##
```

```
## Perform classical LDA and plot the data, 0.975 tolerance ellipses
##  and LDA separation line
##
x <- pottery[,c("MG", "CA")]
grp <- pottery$origin
lda <- LdaClassic(x, grp)
lda
e1 <- getEllipse(loc=lda@center[1,], cov=lda@cov)
e2 <- getEllipse(loc=lda@center[2,], cov=lda@cov)

plot(CA~MG, data=pottery, col=gcol, pch=gpch,
    xlim=c(min(MG,e1[,1], e2[,1]), max(MG,e1[,1], e2[,1])),
    ylim=c(min(CA,e1[,2], e2[,2]), max(CA,e1[,2], e2[,2])))

ab <- lda@ldf[1,] - lda@ldf[2,]
cc <- lda@ldfconst[1] - lda@ldfconst[2]
abline(a=-cc/ab[2], b=-ab[1]/ab[2], col=2, lwd=2)

lines(e1, type="l", col=col[1])
lines(e2, type="l", col=col[2])

##
## Perform robust (MCD) LDA and plot data, classical and
##  robust separation line
##
plot(CA~MG, data=pottery, col=gcol, pch=gpch)
lda <- LdaClassic(x, grp)
ab <- lda@ldf[1,] - lda@ldf[2,]
cc <- lda@ldfconst[1] - lda@ldfconst[2]
abline(a=-cc/ab[2], b=-ab[1]/ab[2], col=2, lwd=2)
abline(a=-cc/ab[2], b=-ab[1]/ab[2], col=4, lwd=2)

rlda <- Linda(x, grp, method="mcd")
rlda
ab <- rlda@ldf[1,] - rlda@ldf[2,]
cc <- rlda@ldfconst[1] - rlda@ldfconst[2]
abline(a=-cc/ab[2], b=-ab[1]/ab[2], col=2, lwd=2)
```

`PredictLda-class` *Class "PredictLda" - prediction of "Lda" objects*

Description

The prediction of a "Lda" object

Objects from the Class

Objects can be created by calls of the form `new("PredictLda",...)` but most often by invoking 'predict' on a "Lda" object. They contain values meant for printing by 'show'

Slots

`classification`: a factor variable containing the classification of each object

`posterior`: a matrix containing the posterior probabilities

`x`: matrix with the discriminant scores

`ct`: re-classification table of the training sample

Methods

show `signature(object = "PredictLda")`: Prints the results

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

See Also

`Lda-class`

Examples

```
showClass("PredictLda")
```

`PredictQda-class` *Class "PredictQda" - prediction of "Qda" objects*

Description

The prediction of a "Qda" object

Objects from the Class

Objects can be created by calls of the form `new("PredictQda",...)` but most often by invoking 'predict' on a "Qda" object. They contain values meant for printing by 'show'

Slots

`classification`: a factor variable containing the classification of each object

`posterior`: a matrix containing the posterior probabilities

`x`: matrix with the discriminant scores

`ct`: re-classification table of the training sample

Methods

show `signature(object = "PredictQda")`: prints the results

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

Qda-class

Examples

```
showClass("PredictQda")
```

Qda-class *Class "Qda" - virtual base class for all classic and robust QDA classes*

Description

The class `Qda` serves as a base class for deriving all other classes representing the results of classical and robust Quadratic Discriminant Analisys methods

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`call`: the (matched) function call.

`prior`: prior probabilities used, default to group proportions

`counts`: number of observations in each class

`center`: the group means

`cov`: the group covariance matrices

`covinv`: the inverse of the group covariance matrices

`covdet`: the determinants of the group covariance matrices

`method`: a character string giving the estimation method used

`X`: the training data set (same as the input parameter x of the constructor function)

`grp`: grouping variable: a factor specifying the class for each observation.

`control`: object of class `"CovControl"` specifying which estimate and with what estimation options to use for the group means and covariances (or `NULL` for classical discriminant analysis)

Methods

predict `signature(object = "Qda")`: calculates prediction using the results in `object`. An optional data frame or matrix in which to look for variables with which to predict. If omitted, the scores are used. If the original fit used a formula or a data frame or a matrix with column names, newdata must contain columns with the same names. Otherwise it must contain the same number of columns, to be used in the same order.

show `signature(object = "Qda")`: prints the results

summary `signature(object = "Qda")`: prints summary information

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

QdaClassic, QdaClassic-class, QdaRobust-class

Examples

```
showClass("Qda")
```

QdaClassic *Quadratic Discriminant Analysis*

Description

Performs quadratic discriminant analysis and returns the results as an object of class `QdaClassic` (aka constructor).

Usage

```
QdaClassic(x, ...)


## Default S3 method:
QdaClassic(x, grouping, prior = proportions, tol = 1.0e-4, ...)
```

Arguments

x	a matrix or data frame containing the explanatory variables (training set).
grouping	grouping variable: a factor specifying the class for each observation.
prior	prior probabilities, default to the class proportions for the training set.
tol	tolerance
...	arguments passed to or from other methods.

Value

Returns an S4 object of class `QdaClassic`

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

Qda-class, QdaClassic-class,

QdaClassic-class *Class "QdaClassic" - Quadratic Discriminant Analysis*

Description

Contains the results of classical Quadratic Discriminant Analysis

Objects from the Class

Objects can be created by calls of the form `new("QdaClassic",...)` but the usual way of creating `QdaClassic` objects is a call to the function `QdaClassic` which serves as a constructor.

Slots

`call`: The (matched) function call.

`prior`: Prior probabilities used, default to group proportions

`counts`: number of observations in each class

`center`: the group means

`cov`: the group covariance matrices

`covinv`: the inverse of the group covariance matrices `covdet`: the determinants of the group covariance matrices

`method`: a character string giving the estimation method used

`X`: the training data set (same as the input parameter x of the constructor function)

`grp`: grouping variable: a factor specifying the class for each observation.

`control`: Object of class "CovControl" inherited from class `Qda` specifying which estimate and with what estimation options to use for the group means and covariances. It is always `NULL` for classical discriminant analysis.

Extends

Class "`Qda`", directly.

Methods

No methods defined with class "QdaClassic" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

`QdaRobust-class`, `Qda-class`, `QdaClassic`

Examples

```
showClass("QdaClassic")
```

QdaCov *Robust Quadratic Discriminant Analysis*

Description

Performs robust quadratic discriminant analysis and returns the results as an object of class `QdaCov` (aka constructor).

Usage

```
QdaCov(x, ...)

## Default S3 method:
QdaCov(x, grouping, prior = proportions, tol = 1.0e-4,
                 method = CovControlMcd(), ...)
```

Arguments

`x`	a matrix or data frame containing the explanatory variables (training set).
`grouping`	grouping variable: a factor specifying the class for each observation.
`prior`	prior probabilities, default to the class proportions for the training set.
`tol`	tolerance
`method`	method
`...`	arguments passed to or from other methods

Details

details

Value

Returns an S4 object of class `QdaCov`

Warning

Still an experimental version!

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

CovMcd

Examples

```
## Example anorexia
library(MASS)
data(anorexia)

## start with the classical estimates
qda <- QdaClassic(Treat~., data=anorexia)
predict(qda)@classification

## try now the robust LDA with the default method (MCD with pooled whitin cov matrix)
rqda <- QdaCov(Treat~., data= anorexia)
predict(rqda)@classification

## try the other methods
QdaCov(Treat~., data= anorexia, method="sde")
```

```
QdaCov(Treat~., data= anorexia, method="M")
QdaCov(Treat~., data= anorexia, method=CovControlOgk())
```

QdaCov-class *Class "QdaCov" - Robust methods for Quadratic Discriminant Analysis*

Description

Robust quadratic discriminant analysis is performed by replacing the classical group means and withing group covariance matrices by their robust equivalents.

Objects from the Class

Objects can be created by calls of the form `new("QdaCov",...)` but the usual way of creating `QdaCov` objects is a call to the function `QdaCov` which serves as a constructor.

Slots

`call`: The (matched) function call.

`prior`: Prior probabilities used, default to group proportions

`counts`: number of observations in each class

`center`: the group means

`cov`: the group covariance matrices

`covinv`: the inverse of the group covariance matrices

`covdet`: the determinants of the group covariance matrices

`method`: a character string giving the estimation method used

`X`: the training data set (same as the input parameter x of the constructor function)

`grp`: grouping variable: a factor specifying the class for each observation.

`control`: Object of class `"CovControl"` specifying which estimate to use for the group means and covariances

Extends

Class "`QdaRobust`", directly. Class "`Qda`", by class "QdaRobust", distance 2.

Methods

No methods defined with class "QdaCov" in the signature.

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

QdaRobust-class, Qda-class, QdaClassic, QdaClassic-class

Examples

```
showClass("QdaCov")
```

QdaRobust-class *Class "QdaRobust" is a virtual base class for all robust QDA classes*

Description

The class `QdaRobust` searves as a base class for deriving all other classes representing the results of robust Quadratic Discriminant Analysis methods

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`call`: The (matched) function call.

`prior`: Prior probabilities used, default to group proportions

`counts`: number of observations in each class

`center`: the group means

`cov`: the group covariance matrices

`covinv`: the inverse of the group covariance matrices

`covdet`: the determinants of the group covariance matrices

`method`: a character string giving the estimation method used

`X`: the training data set (same as the input parameter x of the constructor function)

`grp`: grouping variable: a factor specifying the class for each observation.

`control`: Object of class `"CovControl"` specifying which estimate to use for the group means and covariances

Extends

Class `"Qda"`, directly.

Methods

No methods defined with class "QdaRobust" in the signature.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

Qda-class, QdaClassic-class,

Examples

```
showClass("QdaRobust")
```

---

restimate-methods *Methods for Function estimate in Package 'rrcov'*

Description

Each concrete control class, like `CovControlMest`, `CovControlOgk`, etc., should implement an `restimate` method which will call the correponding (constructor)-function and will return the obtained S4 class, derived from `CovRobust`.

Usage

```
## S4 method for signature 'CovControlMest'
restimate(obj, x, ...)
```

Arguments

`obj`	an object of class "CovControlEstimate"
`x`	Data frame or matrix containing the data .
`...`	other parameters to be passed through to the estimation function.

Methods

obj = "CovControlMcd" Compute the MCD estimates of multivariate location and scatter by callingCovMcd

obj = "CovControlMest" Compute the constrained M-estimates of multivariate location and scatter by callingCovMest

obj = "CovControlOgk" Compute the Ortogonalized Gnanadesikan-Kettenring (OGK) estimates of multivariate location and scatter by callingCovOgk

rice — *Rice taste data*

Description

The rice taste data consists of five inputs and a single output whose values are associated with subjective evaluations as follows: xl: flavor, x2: appearance, x3: taste, x4: stickiness, x5: toughness, y: overall evaluation. Sensory test data have been obtained by such subjective evaluations for 105 kinds of rice (e.g., Sasanishiki, Akita-Komachi, etc.). The data set was used by Nozaki et al. (1997) to demonstrate the high performance of a proposed for automatically generating fuzzy if-then rules from numerical data.

Usage

```
data(rice)
```

Format

A data frame with 105 observations on the following 6 variables:

`Favor` compactness

`Appearance` circularity

`Taste` distance circularity

`Stickiness` radius ratio

`Toughness` principal axis aspect ratio

`Overall_evaluation` maximum length aspect ratio

Source

Nozaki, K., Ishibuchi, H, and Tanaka, H. (1997) A simple but powerful heuristic method for generating fuzzy rules from numerical data *Fuzzy Sets and Systems* **86** 3 p. 251–270.

salmon — *Salmon data*

Description

The salmon data contains two measurements of the growth rings on the scale of Alaskan and Canadian salmon as well as the gender of the fishes. There are 50 Alaskan-born and 50 Canadian-born salmon, and this information is coded in the variable `Origin`.

Usage

```
data(salmon)
```

Format

A data frame with 100 observations on the following 4 variables.

`Gender` female=1 and male=2

`Freshwater` diameter of rings for the first-year freshwater growth (hundrets of an inch)

`Marine` diameter of rings for the first-year marine growth (hundrets of an inch)

`Origin` Origin of the fish: a factor with levels `Alaskan` `Canadian`

Source

Johnson, R.A. and Wichern, D. W. *Applied Multivariate Statistical Analysis* (Prentice Hall, International Editions, 2002, fifth edition)

Examples

```
data(salmon)
```

---

scorePlot-methods *Score plot for Principal Components (objects of class 'Pca')*

Description

Produces a score plot from an object (derived from) `Pca-class`.

Usage

```
    ## S4 method for signature 'Pca'
scorePlot(x, i=1, j=2, ...)
```

Arguments

`x`	an object of class (derived from) `"Pca"`.
`i`	First score coordinate, defaults to `i=1`.
`j`	Second score coordinate, defaults to `j=2`.
`...`	optional arguments to be passed to the internal graphical functions.

Side Effects

a plot is produced on the current graphics device.

Methods

scorePlot `signature(x = Pca)`: Plot a scatter plot of ith against jth score of the Pca object with superimposed tollerance (0.975) ellipse. See also `biplot`, `screeplot`.

See Also

Pca-class, PcaClassic, PcaRobust-class.

Examples

```
require(graphics)

## PCA of the Hawkins Bradu Kass's Artificial Data
##  using all 4 variables
data(hbk)
pca <- PcaHubert(hbk)
pca

scorePlot(pca)
```

---

soil *Exchangable cations in forest soil data set*

Description

The forest soil data set contains measurements on 58 soil pits in the Hubbard Brook Experimental Forest in north-central New Hampshire. The excavations were done in 1983 and 1986. The soil samples were analyzed for the exchangeable cations of aluminium, calcium, magnesium, potassium and sodium. The pit locations in both data sets can be classified by the type of the forest:

- 1: spruce-fir (11 samples),
- 2: high elevation hardwood (23 samples) and
- 3: low elevation hardwood (24 samples)).

Additionally the degree of logging disturbance can be considered (all 0 in the 1983 data set):

- 0: uncut forest,
- 1: cut, undisturbed by machinery and
- 2: cut, disturbed.

The observations are expressed in grams of exchangeable cations per square meter.

Usage

```
data(soil)
```

Format

A data frame with 116 observations on the following 7 variables.

- `F` Type of forest
- `D` Degree of logging disturbance
- `Al` Level of the exchangable cations in Al
- `Ca` Level of the exchangable cations in Ca
- `Mg` Level of the exchangable cations in Mg
- `K` Level of the exchangable cations in K
- `Na` Level of the exchangable cations in Na

Source

Morrison D.F., 2005, Multivariate Statistical Methods, Thompson

References

Vanden Branden K, Hubert M (2005). Robust Classiffication in High Dimensions Based on the SIMCA Method. *Cbemometrics and Intelligent Laboratoty Sysiem*, 79: 10–21.

Examples

```
data(soil)
soil1983 <- soil[soil$D == 0, -2]       # only 1983, remove column D (always 0)

(cc <- Linda(F~., data=soil))
(pr <- predict(cc))
pr@classification
```

SummaryCov-class *Class "SummaryCov" - summary of "Cov" objects*

Description

The "Cov" object plus some additional summary information

Objects from the Class

Objects can be created by calls of the form `new("SummaryCov",...)`, but most often by invoking 'summary' on a "Cov" object. They contain values meant for printing by 'show'.

Slots

- `covobj`: Object of class `"Cov"`
- `evals`: eigenvalues of the covariance or correlation matrix

Methods

getCenter `signature(obj = "SummaryCov")`: location vector

getCov `signature(obj = "SummaryCov")`: covariance matrix

getDistance `signature(obj = "SummaryCov")`: vector of distances

getEvals `signature(obj = "SummaryCov")`: vector of eignevalues

isClassic `signature(obj = "SummaryCov")`: is the estimate a classic one

show `signature(object = "SummaryCov")`: display the object

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

Cov-class

Examples

```
showClass("SummaryCov")
```

`SummaryCovRobust-class`
Class "SummaryCovRobust" - summary of "CovRobust" objects

Description

Summary information for CovRobust objects meants for printing by 'show'

Objects from the Class

Objects can be created by calls of the form `new("SummaryCovRobust",...)`, but most often by invoking 'summary' on an "Cov" object. They contain values meant for printing by 'show'.

Slots

`covobj`**:** Object of class `"Cov"`

`evals`**:** Eigenvalues of the covariance or correlation matrix

Extends

Class `"SummaryCov"`, directly.

Methods

show `signature(object = "SummaryCovRobust")`: ...

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

CovRobust-class, SummaryCov-class

Examples

```
data(hbk)
hbk.x <- data.matrix(hbk[, 1:3])
cv <- CovMest(hbk.x)
cv
summary(cv)
```

SummaryLda-class *Class "SummaryLda" - summary of "Lda" objects*

Description

Contains summary information about an `Lda` object - Linear Discriminant Analysis object

Objects from the Class

Objects can be created by calls of the form `new("SummaryLda",...)`, but most often by invoking 'summary' on an "Lda" object. They contain values meant for printing by 'show'.

Slots

`ldaobj`**:** Object of class `"Lda"`

Methods

show `signature(object = "SummaryLda")`: display the object

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

Lda-class

Examples

```
showClass("SummaryLda")
```

SummaryPca-class *Class "SummaryPca" - summary of "Pca" objects*

Description

The "Pca" object plus some additional summary information

Objects from the Class

Objects can be created by calls of the form `new("SummaryPca",...)`, but most often by invoking 'summary' on a "Pca" object. They contain values meant for printing by 'show'.

Slots

`pcaobj`**:** Object of class `"Pca"`

`importance`**:** matrix with additional information: importance of components

Methods

show `signature(object = "SummaryPca")`: display the object

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL http://www.jstatsoft.org/v32/i03/.

See Also

Pca-class

Examples

```
showClass("SummaryPca")
```

SummaryQda-class   *Class "SummaryQda" - summary of "Qda" objects*

Description

Summary information about a `Qda` - Quadratic Discriminant Analysis object

Objects from the Class

Objects can be created by calls of the form `new("SummaryQda",...)`, but most often by invoking 'summary' on an "Qda" object. They contain values meant for printing by 'show'.

Slots

`qdaobj`: Object of class `"Qda"`

Methods

show `signature(object = "SummaryQda")`: display the object

Author(s)

Valentin Todorov `<valentin.todorov@chello.at>`

References

Todorov V & Filzmoser P (2009), An Object Oriented Framework for Robust Multivariate Analysis. *Journal of Statistical Software*, **32**(3), 1–47. URL `http://www.jstatsoft.org/v32/i03/`.

See Also

`Qda-class`

Examples

```
showClass("SummaryQda")
```

T2.test *Robust Hotelling T2 test*

Description

Performs one and two sample Hotelling T2 tests as well as robust one-sample Hotelling T2 test

Usage

```
T2.test(x, ...)

## Default S3 method:
T2.test(x, y = NULL, mu = 0, conf.level = 0.95, method=c("c", "mcd"), ...)

## S3 method for class 'formula'
T2.test(formula, data, subset, na.action, ...)
```

Arguments

`x`	a (non-empty) numeric data frame or matrix.
`y`	an optional (non-empty) numeric data frame or matrix.
`mu`	an optional (non-empty) numeric vector of data values (or a single number which will be repeated p times) indicating the true value of the mean (or difference in means if you are performing a two sample test).
`conf.level`	confidence level of the interval
`method`	the method to be used - 'c' for sample mean and covariance matrix and 'mcd' for minimum covariance determinant estimator. A two-sample MCD based T2-test is not yet implemented.
`formula`	a formula of the form `lhs ~ rhs` where `lhs` is a numeric data frame or matrix giving the observations and `rhs` a factor with two levels giving the corresponding groups.
`data`	an optional matrix or data frame (or similar: see `model.frame`) containing the variables in the formula `formula`. By default the variables are taken from `environment(formula)`.
`subset`	an optional vector specifying a subset of observations to be used (currently not used)
`na.action`	a function which indicates what should happen when the data contain `NA`s. Defaults to `getOption("na.action")` (currently only "na.rm" used)
`...`	further arguments to be passed to or from methods.

Details

The formula interface is only applicable for the two-sample tests.

Value

A list with class "htest" containing the following components:

statistic	the value of the T2-statistic.
parameter	the degrees of freedom for the T2-statistic.
p.value	the p-value for the test.
conf.int	a confidence interval for the mean vector appropriate to the specified alternative hypothesis.
estimate	the estimated mean vector or vectors depending on whether it was a one-sample test or a two-sample test.
null.value	the specified hypothesized value of the mean or mean difference depending on whether it was a one-sample test or a two-sample test.
alternative	a character string describing the alternative hypothesis.
method	a character string indicating what type of T2-test was performed.
data.name	a character string giving the name(s) of the data.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Willems G., Pison G., Rousseeuw P. and Van Aelst S. (2002), A robust hotelling test, *Metrika*, **55**, 125–138.

See Also

CovMcd

Examples

```
## One-sample classical test
data(delivery)
delivery.x <- delivery[,1:2]
T2.test(delivery.x)

## One-sample robust test
data(delivery)
delivery.x <- delivery[,1:2]
T2.test(delivery.x, method="mcd")

## Two-sample classical test
data(hemophilia)
grp <-as.factor(hemophilia[,3])
x <- hemophilia[which(grp==levels(grp)[1]),1:2]
y <- hemophilia[which(grp==levels(grp)[2]),1:2]
T2.test(x,y)
```

```
## or using the formula interface
T2.test(as.matrix(hemophilia[,-3])~hemophilia[,3])


## Not run:
## Two-sample robust test
T2.test(x,y, method="mcd")    ## error - not yet implemented

## End(Not run)
```

---

un86 *United Nations Data - 1986*

Description

This data set consists of seven socioeconomic variables observed for 73 countries.

Usage

```
data(un86)
```

Format

A data frame with 73 observations on the following 7 variables.

POP Total population in milions

MOR Number of infant deaths per thousand births

CAR Number of motorized vehicles per hundred inhabitants

DR Number of medical doctors per thousand inhabitants

GNP Gross national product per inhabitant in thousands of US dollars

DEN Density in inhabitants per square kilometer

TB Trade balance, defined as total exports/(total exports + total imports)

Details

The data set is from World Statistics in Brief, Number 10, a 1986 UN publication. It was used in Daigle et al. (1992) to illustrate a robust biplot method.

Source

World Statistics in Brief, Number 10, a 1986 United Nations publication

Daigle, G. and Rivest, L. (1992) A robust biplot, The canadian Journal of Statistics, 20, pp 241–255

Examples

```
data(un86)
pairs(un86)
```

`wages` *Wages and Hours*

Description

The data are from a national sample of 6000 households with a male head earning less than USD 15,000 annually in 1966. The data were clasified into 39 demographic groups for analysis. The study was undertaken in the context of proposals for a guaranteed annual wage (negative income tax). At issue was the response of labor supply (average hours) to increasing hourly wages. The study was undertaken to estimate this response from available data.

Usage

```
data(wages)
```

Format

A data frame with 39 observations on the following 10 variables:

- `HRS` Average hours worked during the year
- `RATE` Average hourly wage (USD)
- `ERSP` Average yearly earnings of spouse (USD)
- `ERNO` Average yearly earnings of other family members (USD)
- `NEIN` Average yearly non-earned income
- `ASSET` Average family asset holdings (Bank account, etc.) (USD)
- `AGE` Average age of respondent
- `DEP` Average number of dependents
- `RACE` Percent of white respondents
- `SCHOOL` Average highest grade of school completed

Source

DASL library

'http://lib.stat.cmu.edu/DASL/Datafiles/wagesdat.html'

References

D.H. Greenberg and M. Kosters, (1970). Income Guarantees and the Working Poor, The Rand Corporation.

Examples

```
data(wages)
names(wages)
x <- as.matrix(wages)
ok <- is.finite(x %*% rep(1, ncol(x)))
wages <- wages[ok, , drop = FALSE]
wages.lm <- lm(HRS~AGE, data=wages)
plot(HRS ~ AGE, data = wages)
abline(wages.lm)
class(wages.lm)
names(wages.lm)
summary(wages.lm)

wages.mm <- lmrob(HRS~AGE, data=wages)
plot(HRS ~ AGE, data = wages)
abline(wages.mm)
class(wages.mm)
names(wages.mm)
summary(wages.mm)
```

Wilks.test *Classical and Robust One-way MANOVA: Wilks Lambda*

Description

Classical and Robust One-way MANOVA: Wilks Lambda

Usage

```
## S3 method for class 'formula'
Wilks.test(formula, data, ..., subset, na.action)

## Default S3 method:
Wilks.test(x, grouping, method=c("c", "mcd", "rank"),
    approximation=c("Bartlett", "Rao", "empirical"),
    xd=NULL, xq=NULL, xfn = NULL, xwl=NULL, nrep=3000, trace=FALSE, ...)

## S3 method for class 'data.frame'
Wilks.test(x, ...)

## S3 method for class 'matrix'
Wilks.test(x, grouping, ..., subset, na.action)
```

Arguments

formula A formula of the form `groups ~ x1 + x2 + ...` That is, the response is the grouping factor and the right hand side specifies the (non-factor) variables.

`data`	Data frame from which variables specified in `formula` are to be taken.
`x`	(required if no formula is given as the principal argument.) a matrix or data frame or Matrix containing the explanatory variables.
`grouping`	grouping variable - a factor specifying the class for each observation (required if no formula argument is given.)
`subset`	An index vector specifying the cases to be used.
`na.action`	A function to specify the action to be taken if `NA`s are found. The default action is for the procedure to fail. An alternative is `na.omit`, which leads to rejection of cases with missing values on any required variable.
`method`	`"c"` for standard estimators of the mean and variance, `"mcd"` for MCD estimators of mean and variances and `"rank"` for rank based wilks' lambda as proposed by Nath and Pavur (1985).
`approximation`	`"Bartlett"` for Bartlett approximation (default), `"Rao"` for rao approximation (only for method="c") and `"empirical"` for simulated empirical distribution.
`xd`	multiplication factor for the approximate distribution of the robust Lambda statistic. If `xd=NULL` the factor will computed by simulation and will be returned in the value (see Details)
`xq`	the degrees of freedom for the approximate χ^2 distribution of the robust Lambda statistic. If `xq=NULL` the degrees of freedom will computed by simulation and will be returned in the value (see Details)
`xfn`	the empirical distribution function. If `xfn=NULL` the empirical function will be estimated by simulation and will be returned in the value (see Details)
`xwl`	the simulated values of the robust statistic. If `xwl=NULL` the simulation will be performed and the calculated result will be returned in the value (see Details)
`nrep`	number of trials for the simulations for computing the multiplication factor `xd` and the degrees of freedom `xq`. Default is `nrep=3000`.
`trace`	whether to print intermediate results. Default is `trace = FALSE`
`...`	arguments passed to or from other methods.

Details

The classical Wilks' Lambda statistic for testing the equality of the group means of two or more groups is modified into a robust one through substituting the classical estimates by the highly robust and efficient reweighted MCD estimates, which can be computed efficiently by the FAST-MCD algorithm - see `CovMcd`. An approximation for the finite sample distribution of the Lambda statistic is obtained, based on matching the mean and variance of a multiple of an χ^2 distribution which are computed by simultaion.

Value

A list with class `"htest"` containing the following components:

`statistic`	the value of the Wilks' Lambda statistic.
`parameter`	The corresponding approximation of the Wilks' lambda statistic and the degrees of freedom.

`p.value`	the p-value for the test.
`estimate`	the estimated mean vectors.
`method`	a character string indicating what type of test was performed.
`data.name`	a character string giving the name of the data.
`xd`	multiplication factor for the approximate distribution of the robust Lambda statistic.
`xq`	the degrees of freedom for the approximate χ^2 distribution of the robust Lambda statistic.

Note

This function may be called giving either a formula and optional data frame, or a matrix and grouping factor as the first two arguments. All other arguments are optional.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Todorov, V. and Filzmoser, P. (2007) Robust statistic for the one-way MANOVA, *submetted to the Journal of Environmetrics*.

Todorov, V. (2007) Robust selection of variables in linear discriminant analysis, *Statistical Methods and Applications*, **15**, 395.407, doi:10.1007/s10260-006-0032-6.

Nath, R. and Pavur, R. (1985) A new statistic in the one way multivariate analysis of variance, *Computatational Statistics and Data Analysis*, **2**, 297–315

See Also

`CovMcd`, `T2.test`

Examples

```
library(MASS)
data(anorexia)
grp <- as.factor(anorexia[,1])
x <- as.matrix(anorexia[,2:3])
##  Using the default interface, classical test
Wilks.test(x, grouping=grp, method="c")

##  Using the default interface, rank based test
Wilks.test(x, grouping=grp, method="rank")

## For this data set: p=2, n=n1+n2+n3=29+26+17
## were computed the following multiplication factor xd and degrees of freedom xq
##  for the MCD estimates with alpha=0.5
xd <-  -0.02162666
xq <- 3.63971
Wilks.test(x, grouping=grp, method="mcd", xd=xd, xq=xq)
```

```
## Now the same with the formula interface
Wilks.test(Treat~Prewt+Postwt, data=anorexia, method="mcd", xd=xd, xq=xq)

##Iris data with formula interface
data(iris)
Wilks.test(Species~., data=iris, method="c")

## and with default interface
Wilks.test(iris[,1:4],grouping=iris[,5], method="c")

# hemophilia data - classical, rank and MCD test
data(hemophilia)
hemophilia$gr <- as.factor(hemophilia$gr)

Wilks.test(gr~., data=hemophilia, method="c")
Wilks.test(gr~., data=hemophilia, method="rank")
## already simulated parameters for MCD with alpha=0.5
xd <- -0.01805436
xq <- 1.950301
Wilks.test(gr~., data=hemophilia, xd=xd, xq=xq, method="mcd")
```

Index