Package 'rrcovHD'

January 27, 2015

Date 2014-12-02

Title Robust multivariate Methods for High Dimensional Data

Version 0.2-3

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

Description Robust multivariate methods for high dimensional data: outlier detection, PCA, PLS, classification

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

Depends rrcov (>= 1.3-7), robustbase (>= 0.92-1), methods

Imports pls, spls, pcaPP

LazyLoad yes

LazyData yes

License GPL (>= 2)

NeedsCompilation no

Repository CRAN

Date/Publication 2014-12-02 16:36:38

R topics documented:

`Cars` *Consumer reports car data: dimensions*

Description

A data frame containing 11 variables with different dimensions of 111 cars

Usage

```
data(Cars)
```

Format

A data frame with 111 observations on the following 11 variables.

`length` a numeric vector

`wheelbase` a numeric vector

`width` a numeric vector

`height` a numeric vector

`front.hd` a numeric vector

`rear.hd` a numeric vector

`front.leg` a numeric vector

`rear.seating` a numeric vector

`front.shoulder` a numeric vector

`rear.shoulder` a numeric vector

`luggage` a numeric vector

Source

Consumer reports. (April 1990). http://backissues.com/issue/Consumer-Reports-April-1990, pp. 235–288.

References

Chambers, J. M. and Hastie, T. J. (1992). Statistical models in S. Cole, Pacific Grove, CA: Wadsworth and Brooks, pp. 46–47.

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

Examples

```
    data(Cars)

## Plot a pairwise scaterplot matrix
    pairs(Cars[,1:6])

    mcd <- CovMcd(Cars[,1:6])
    plot(mcd, which="pairs")

## Start with robust PCA
    pca <- PcaHubert(Cars, kmax=11)
    pca

## Compare with the classical PCA
    prcomp(Cars)

## or
    PcaClassic(Cars, kmax=11)

## If you want to print the scores too, use
    print(pca, print.x=TRUE)

## Using the formula interface
    PcaHubert(~., data=Cars, kmax=11)

## To plot the results:

    plot(pca)                   # distance plot
    pca2 <- PcaHubert(Cars, k=4)
    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(Cars, kmax=11)
    cov.1 <- py@loadings %*% diag(py@eigenvalues) %*% t(py@loadings)
    cov.1
```

CSimca *Classification in high dimensions based on the (classical) SIMCA method*

Description

CSimca performs the (classical) SIMCA method. This method classifies a data matrix x with a known group structure. To reduce the dimension on each group a PCA analysis is performed. Afterwards a classification rule is developped to determine the assignment of new observations.

Usage

```
CSimca(x, ...)
## Default S3 method:
CSimca(x, grouping, prior=proportions, k, kmax = ncol(x),
    tol = 1.0e-4, trace=FALSE, ...)
## S3 method for class 'formula'
CSimca(formula, data = NULL, ..., subset, na.action)
```

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
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.
trace	whether to print intermediate results. Default is trace = FALSE
...	arguments passed to or from other methods.

Details

CSimca, serving as a constructor for objects of class CSimca-class is a generic function with "formula" and "default" methods.

SIMCA is a two phase procedure consisting of PCA performed on each group separately for dimension reduction followed by classification rules built in the lower dimensional space (note that the dimension in each group can be different). In original SIMCA new observations are classified by means of their deviations from the different PCA models. Here (and also in the robust versions implemented in this package) the classification rules will be obtained using two popular distances arising from PCA - orthogonal distances (OD) and score distances (SD). For the definition of these distances, the definition of the cutoff values and the standartization of the distances see Vanden Branden K, Hubert M (2005) and Todorov and Filzmoser (2009).

Value

An S4 object of class CSimca-class which is a subclass of of the virtual class Simca-class.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Vanden Branden K, Hubert M (2005) Robust classification in high dimensions based on the SIMCA method. Chemometrics and Intellegent Laboratory Systems 79:10–21

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(pottery)
dim(pottery)        # 27 observations in 2 classes, 6 variables
head(pottery)

## Build the SIMCA model. Use RSimca for a robust version
cs <- CSimca(origin~., data=pottery)
cs
summary(cs)


## generate a sample from the pottery data set -
##  this will be the "new" data to be predicted
smpl <- sample(1:nrow(pottery), 5)
test <- pottery[smpl, -7]        # extract the test sample. Remove the last (grouping) variable
print(test)


## predict new data
pr <- predict(cs, newdata=test)

pr@classification
```

CSimca-class *Class "CSimca" - classification in high dimensions based on the (classical) SIMCA method*

Description

The class CSimca represents the SIMCA algorithm for classification in high dimensions. The objects of class CSImca contain the results of the SIMCA method.

Objects from the Class

Objects can be created by calls of the form new("CSImca", ...) but the usual way of creating CSimca objects is a call to the function CSimca() 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

pcaobj: A list of Pca objects - one for each group

k: Object of class "numeric" number of (choosen) principal components

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

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 "Simca", directly.

Methods

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

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Vanden Branden K, Hubert M (2005) Robust classification in high dimensions based on the SIMCA method. Chemometrics and Intellegent Laboratory Systems 79:10–21

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("CSimca")
```

`getWeight-methods` *Accessor methods to the essential slots of* `Outlier` *and its subclasses*

Description

Accessor methods to the essential slots of `Outlier` and its subclasses

Methods

obj = "Outlier" generic functions - see `getWeight`, `getOutliers`, `getClassLabels`, `getCutoff`

`kibler` *1985 Auto Imports Database*

Description

The original data set `kibler.orig` consists of three types of entities: (a) the specification of an auto in terms of various characteristics, (b) its assigned insurance risk rating and (c) its normalized losses in use as compared to other cars.

The second rating corresponds to the degree to which the auto is more risky than its price indicates. Cars are initially assigned a risk factor symbol associated with its price. Then, if it is more risky (or less), this symbol is adjusted by moving it up (or down) the scale. Actuarians call this process "symboling". A value of +3 indicates that the auto is risky, -3 that it is probably pretty safe.

The third factor is the relative average loss payment per insured vehicle year. This value is normalized for all autos within a particular size classification (two-door small, station wagons, sports/speciality, etc...), and represents the average loss per car per year.

Usage

```
data(kibler)
```

Format

A data frame with 195 observations on the following 14 variables. The original data set (also available as `kibler.orig`) contains 205 cases and 26 variables of which 15 continuous, 1 integer and 10 nominal. The non-numeric variables and variables with many missing values were removed. Cases with missing values were removed too.

`symboling` a numeric vector

`wheel-base` a numeric vector

`length` a numeric vector

`width` a numeric vector

`height` a numeric vector

`curb-weight` a numeric vector

`bore` a numeric vector

`stroke` a numeric vector

`compression-ratio` a numeric vector

`horsepower` a numeric vector

`peak-rpm` a numeric vector

`city-mpg` a numeric vector

`highway-mpg` a numeric vector

`price` a numeric vector

Details

The original data set contains 205 cases and 26 variables of which 15 continuous, 1 integer and 10 nominal. The non-numeric variables and variables with many missing values were removed. Cases with missing values were removed too. Thus the data set remains with 195 cases and 14 variables.

Source

www.cs.umb.edu/~rickb/files/UCI/

References

Kibler, D., Aha, D.W. and Albert, M. (1989). Instance-based prediction of real-valued attributes. *Computational Intelligence*, Vo.l 5, 51-57.

Examples

```
data(kibler)
x.sd <- apply(kibler,2,sd)
xsd <- sweep(kibler, 2, x.sd, "/", check.margin = FALSE)
apply(xsd, 2, sd)

x.mad <- apply(kibler, 2, mad)
xmad <- sweep(kibler, 2, x.mad, "/", check.margin = FALSE)
apply(xmad, 2, mad)

x.qn <- apply(kibler, 2, Qn)
xqn <- sweep(kibler, 2, x.qn, "/", check.margin = FALSE)
apply(xqn, 2, Qn)


## Display the scree plot of the classical and robust PCA
screeplot(PcaClassic(xsd))
screeplot(PcaGrid(xqn))

########################################
```

```
##
## DD-plots
##
## Not run:
usr <- par(mfrow=c(2,2))
plot(SPcaGrid(xsd, lambda=0, method="sd", k=4), main="Standard PCA")    # standard
plot(SPcaGrid(xqn, lambda=0, method="Qn", k=4))                      # robust, non-sparse

plot(SPcaGrid(xqn, lambda=1,43, method="sd", k=4), main="Stdandard sparse PCA")  # sparse
plot(SPcaGrid(xqn, lambda=2.36, method="Qn", k=4), main="Robust sparse PCA")    # robust sparse
par(usr)

########################################
##  Table 2 in Croux et al
##  - to compute EV=Explained variance and Cumulative EV we
##      need to get all 14 eigenvalues
##
rpca <- SPcaGrid(xqn, lambda=0, k=14)
srpca <- SPcaGrid(xqn, lambda=2.36, k=14)
tab <- cbind(round(getLoadings(rpca)[,1:4], 2), round(getLoadings(srpca)[,1:4], 2))

vars1 <- getEigenvalues(rpca);  vars1 <- vars1/sum(vars1)
vars2 <- getEigenvalues(srpca); vars2 <- vars2/sum(vars2)
cvars1 <- cumsum(vars1)
cvars2 <- cumsum(vars2)
ev <- round(c(vars1[1:4], vars2[1:4]),2)
cev <- round(c(cvars1[1:4], cvars2[1:4]),2)
rbind(tab, ev, cev)

## End(Not run)
```

`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 <- CSimca(grp~., data=olitos, k=c(3,4,2,2))
cc
pr <- predict(cc, method=2)
tt <- rrcov::mtxconfusion(cc@grp, pr@classification, printit=TRUE)
```

`Outlier-class` *Class* "`Outlier`" *– a base class for outlier identification*

Description

The class `Outlier` represents the results of outlier identification.

Objects from the Class

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

Slots

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

`counts`: Number of observations in each class

`grp`: Grouping variable

`wt`: Vector of weights

`flag`: 0/1 flags identifying the outliers

`method`: A character string specifying the method used to identify the outliers. In case of `OutlierMahdist` class this is the name of the robust estimator of multivariate location and covariance matrix used

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

Methods

getClassLabels Returns a vector with indices for a given class

getDistance Returns a vector containing the computed distances

getFlag Returns the flags identifying the outliers

getOutliers Returns a vector with the indices of the identified outliers

getWeight Returns a vector of weights

plot

show

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/.

Filzmoser P & Todorov V (2012), Robust tools for the imperfect world, To appear.

Examples

```
showClass("Outlier")
```

`OutlierMahdist` *Outlier identification using robust (mahalanobis) distances based on robust multivariate location and covariance matrix*

Description

This function uses the Mahalanobis distance as a basis for multivariate outlier detection. The standard method for multivariate outlier detection is robust estimation of the parameters in the Mahalanobis distance and the comparison with a critical value of the Chi2 distribution (Rousseeuw and Van Zomeren, 1990).

Usage

```
    OutlierMahdist(x, ...)
    ## Default S3 method:
OutlierMahdist(x, grouping, control, trace=FALSE, ...)
    ## S3 method for class 'formula'
OutlierMahdist(formula, data, ..., 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 matrix or data frame.
`grouping`	grouping variable: a factor specifying the class for each observation.
`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)
`trace`	whether to print intermediate results. Default is `trace = FALSE`

Details

If the data set consists of two or more classes (specified by the grouping variable `grouping`) the proposed method iterates through the classes present in the data, separates each class from the rest and identifies the outliers relative to this class, thus treating both types of outliers, the mislabeled and the abnormal samples in a homogenous way.

The estimation method is selected by 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 - for details see `CovRobust`.

Value

An S4 object of class `OutlierMahdist` which is a subclass of the virtual class `Outlier`.

Author(s)

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

References

P. J. Rousseeuw and B. C. Van Zomeren (1990). Unmasking multivariate outliers and leverage points. *Journal of the American Statistical Association*. Vol. 85(411), pp. 633-651.

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.

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/.

Filzmoser P & Todorov V (2012), Robust tools for the imperfect world, To appear.

Examples

```
data(hemophilia)
obj <- OutlierMahdist(gr~.,data=hemophilia)
obj

getDistance(obj)            # returns an array of distances
getClassLabels(obj, 1)      # returns an array of indices for a given class
getCutoff(obj)           # returns an array of cutoff values (for each class, usually equal)
getFlag(obj)                #  returns an 0/1 array of flags
plot(obj, class=2)          # standard plot function
```

`OutlierMahdist-class` *Class* `OutlierMahdist` - *Outlier identification using robust (mahalanobis) distances based on robust multivariate location and covariance matrix*

Description

Holds the results of outlier identification using robust mahalanobis distances computed by robust multivarite location and covarince matrix.

Objects from the Class

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

Slots

`covobj`: A list containing the robust estimates of multivariate location and covariance matrix for each class

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

`counts`: Number of observations in each class

`grp`: Grouping variable

`wt`: Weights

`flag`: 0/1 flags identifying the outliers

`method`: Method used to compute the robust estimates of multivariate location and covariance matrix

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

Extends

Class `"Outlier"`, directly.

Methods

getCutoff Return the cutoff value used to identify outliers

getDistance Return a vector containing the computed distances

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

OutlierMahdist, Outlier-class

Examples

```
showClass("OutlierMahdist")
```

OutlierPCDist *Outlier identification in high dimensions using the PCDIST algorithm*

Description

The function implements a simple, automatic outlier detection method suitable for high dimensional data that treats each class independently and uses a statistically principled threshold for outliers. The algorithm can detect both mislabeled and abnormal samples without reference to other classes.

Usage

```
    OutlierPCDist(x, ...)
    ## Default S3 method:
OutlierPCDist(x, grouping, control, k, explvar, trace=FALSE, ...)
    ## S3 method for class 'formula'
OutlierPCDist(formula, data, ..., 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 matrix or data frame.
grouping	grouping variable: a factor specifying the class for each observation.
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)
k	Number of components to select for PCA. If missing, the number of components will be calculated automatically

explvar	Minimal explained variance to be used for calculation of the number of components in PCA. If explvar is not provided, automatic dimensionality selection using profile likelihood, as proposed by Zhu and Ghodsi will be used.
trace	whether to print intermediate results. Default is trace = FALSE

Details

If the data set consists of two or more classes (specified by the grouping variable grouping) the proposed method iterates through the classes present in the data, separates each class from the rest and identifies the outliers relative to this class, thus treating both types of outliers, the mislabeled and the abnormal samples in a homogenous way.

The first step of the algorithm is dimensionality reduction using (classical) PCA. The number of components to select can be provided by the user but if missing, the number of components will be calculated either using the provided minimal explained variance or by the automatic dimensionality selection using profile likelihood, as proposed by Zhu and Ghodsi.

Value

An S4 object of class OutlierPCDist which is a subclass of the virtual class Outlier.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

A.D. Shieh and Y.S. Hung (2009), Detecting Outlier Samples in Microarray Data, *Statistical Applications in Genetics and Molecular Biology* Vol. 8.

M. Zhu, and A. Ghodsi (2006). Automatic dimensionality selection from the scree plot via the use of profile likelihood. *Computational Statistics & Data Analysis*, Vol. 51, 918-930.

P. Filzmoser & V. Todorov (2012), Robust tools for the imperfect world, To appear.

See Also

OutlierPCDist, Outlier

Examples

```
data(hemophilia)
obj <- OutlierPCDist(gr~.,data=hemophilia)
obj

getDistance(obj)            # returns an array of distances
getClassLabels(obj, 1)      # returns an array of indices for a given class
getCutoff(obj)           # returns an array of cutoff values (for each class, usually equal)
getFlag(obj)                #  returns an 0/1 array of flags
plot(obj, class=2)          # standard plot function
```

OutlierPCDist-class *Class "OutlierPCDist" - Outlier identification in high dimensions using using the PCDIST algorithm*

Description

The function implements a simple, automatic outlier detection method suitable for high dimensional data that treats each class independently and uses a statistically principled threshold for outliers. The algorithm can detect both mislabeled and abnormal samples without reference to other classes.

Objects from the Class

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

Slots

`covobj`: A list containing intermediate results of the PCDIST algorithm for each class

`k`: Number of selected PC

`call`, `counts`, `grp`, `wt`, `flag`, `method`, `singularity`: from the "`Outlier`" class.

Extends

Class "`Outlier`", directly.

Methods

getCutoff Return the cutoff value used to identify outliers

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

A.D. Shieh and Y.S. Hung (2009), Detecting Outlier Samples in Microarray Data, *Statistical Applications in Genetics and Molecular Biology* Vol. 8.

P. Filzmoser & V. Todorov (2012), Robust tools for the imperfect world, To appear.

See Also

`OutlierPCDist`, `Outlier`

Examples

```
showClass("OutlierPCDist")
```

OutlierPCOut *Outlier identification in high dimensions using the PCOUT algorithm*

Description

The function implements a computationally fast procedure for identifying outliers that is particularly effective in high dimensions. This algorithm utilizes simple properties of principal components to identify outliers in the transformed space, leading to significant computational advantages for high-dimensional data. This approach requires considerably less computational time than existing methods for outlier detection, and is suitable for use on very large data sets. It is also capable of analyzing the data situation commonly found in certain biological applications in which the number of dimensions is several orders of magnitude larger than the number of observations.

Usage

```
    OutlierPCOut(x, ...)
    ## Default S3 method:
OutlierPCOut(x, grouping, explvar=0.99, trace=FALSE, ...)
    ## S3 method for class 'formula'
OutlierPCOut(formula, data, ..., 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 matrix or data frame.
grouping	grouping variable: a factor specifying the class for each observation.
explvar	a numeric value between 0 and 1 indicating how much variance should be covered by the robust PCs (default to 0.99)
trace	whether to print intermediate results. Default is trace = FALSE

Details

If the data set consists of two or more classes (specified by the grouping variable grouping) the proposed method iterates through the classes present in the data, separates each class from the rest and identifies the outliers relative to this class, thus treating both types of outliers, the mislabeled and the abnormal samples in a homogenous way.

Value

An S4 object of class OutlierPCOut which is a subclass of the virtual class Outlier.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

P. Filzmoser, R. Maronna and M. Werner (2008), Outlier identification in high dimensions, *Computational Statistics & Data Analysis*, Vol. 52 1694–1711.

P. Filzmoser & V. Todorov (2012), Robust tools for the imperfect world, To appear.

See Also

OutlierPCOut, Outlier

Examples

```
data(hemophilia)
obj <- OutlierPCOut(gr~.,data=hemophilia)
obj

getDistance(obj)            # returns an array of distances
getClassLabels(obj, 1)      # returns an array of indices for a given class
getCutoff(obj)           # returns an array of cutoff values (for each class, usually equal)
getFlag(obj)                #  returns an 0/1 array of flags
plot(obj, class=2)          # standard plot function
```

OutlierPCOut-class *Class "OutlierPCOut" - Outlier identification in high dimensions using using the PCOUT algorithm*

Description

Holds the results of outlier identification using the PCOUT algorithm.

Objects from the Class

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

Slots

covobj: A list containing intermediate results of the PCOUT algorithm for each class

call, counts, grp, wt, flag, method, singularity: from the "Outlier" class.

Extends

Class "Outlier", directly.

Methods

getCutoff Return the cutoff value used to identify outliers

getDistance Return a vector containing the computed distances

plot Plot the results of the outlier detection process

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

P. Filzmoser, R. Maronna and M. Werner (2008), Outlier identification in high dimensions, *Computational Statistics & Data Analysis*, Vol. 52 1694–1711.

P. Filzmoser & V. Todorov (2012), Robust tools for the imperfect world, To appear.

See Also

OutlierPCOut, "Outlier"

Examples

```
showClass("OutlierMahdist")
```

OutlierSign1 *Outlier identification in high dimensions using the SIGN1 algorithm*

Description

Fast algorithm for identifying multivariate outliers in high-dimensional and/or large datasets, using spatial signs, see Filzmoser, Maronna, and Werner (CSDA, 2007). The computation of the distances is based on Mahalanobis distances.

Usage

```
    OutlierSign1(x, ...)
    ## Default S3 method:
OutlierSign1(x, grouping, qcrit = 0.975, trace=FALSE, ...)
    ## S3 method for class 'formula'
OutlierSign1(formula, data, ..., 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 matrix or data frame.
grouping	grouping variable: a factor specifying the class for each observation.
qcrit	a numeric value between 0 and 1 indicating the quantile to be used as critical value for outlier detection (default to 0.975).
trace	whether to print intermediate results. Default is trace = FALSE

Details

Based on the robustly sphered and normed data, robust principal components are computed. These are used for computing the covariance matrix which is the basis for Mahalanobis distances. A critical value from the chi-square distribution is then used as outlier cutoff.

Value

An S4 object of class OutlierSign1 which is a subclass of the virtual class Outlier.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

P. Filzmoser, R. Maronna and M. Werner (2008), Outlier identification in high dimensions, *Computational Statistics & Data Analysis*, Vol. 52 1694–1711.

P. Filzmoser & V. Todorov (2012), Robust tools for the imperfect world, To appear.

See Also

OutlierSign1, OutlierSign2, Outlier

Examples

```
data(hemophilia)
obj <- OutlierSign1(gr~.,data=hemophilia)
obj

getDistance(obj)           # returns an array of distances
```

```
getClassLabels(obj, 1)      # returns an array of indices for a given class
getCutoff(obj)           # returns an array of cutoff values (for each class, usually equal)
getFlag(obj)                #  returns an 0/1 array of flags
plot(obj, class=2)          # standard plot function
```

`OutlierSign1-class` *Class* "`OutlierSign1`" *- Outlier identification in high dimensions using the SIGN1 algorithm*

Description

Fast algorithm for identifying multivariate outliers in high-dimensional and/or large datasets, using spatial signs, see Filzmoser, Maronna, and Werner (CSDA, 2007). The computation of the distances is based on Mahalanobis distances.

Objects from the Class

Objects can be created by calls of the form `new("OutlierSign1", ...)` but the usual way of creating `OutlierSign1` objects is a call to the function `OutlierSign1()` which serves as a constructor.

Slots

`covobj`**:** A list containing intermediate results of the SIGN1 algorithm for each class

`call`**,** `counts`**,** `grp`**,** `wt`**,** `flag`**,** `method`**,** `singularity`**:** from the "`Outlier`" class.

Extends

Class "`Outlier`", directly.

Methods

getCutoff Return the cutoff value used to identify outliers

getDistance Return a vector containing the computed distances

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

P. Filzmoser, R. Maronna and M. Werner (2008), Outlier identification in high dimensions, *Computational Statistics & Data Analysis*, Vol. 52 1694–1711.

P. Filzmoser & V. Todorov (2012), Robust tools for the imperfect world, To appear.

See Also

`OutlierSign1`, `OutlierSign2`, `Outlier`

Examples

```
showClass("OutlierSign1")
```

OutlierSign2 *Outlier identification in high dimensions using the SIGN2 algorithm*

Description

Fast algorithm for identifying multivariate outliers in high-dimensional and/or large datasets, using spatial signs, see Filzmoser, Maronna, and Werner (CSDA, 2007). The computation of the distances is based on principal components.

Usage

```
    OutlierSign2(x, ...)
    ## Default S3 method:
OutlierSign2(x, grouping, qcrit = 0.975, explvar=0.99, trace=FALSE, ...)
    ## S3 method for class 'formula'
OutlierSign2(formula, data, ..., 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 matrix or data frame.
grouping	grouping variable: a factor specifying the class for each observation.
explvar	a numeric value between 0 and 1 indicating how much variance should be covered by the robust PCs. Default is 0.99.
qcrit	a numeric value between 0 and 1 indicating the quantile to be used as critical value for outlier detection. Default is 0.975.
trace	whether to print intermediate results. Default is trace = FALSE

Details

Based on the robustly sphered and normed data, robust principal components are computed which are needed for determining distances for each observation. The distances are transformed to approach chi-square distribution, and a critical value is then used as outlier cutoff.

Value

An S4 object of class OutlierSign2 which is a subclass of the virtual class Outlier.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

P. Filzmoser, R. Maronna and M. Werner (2008), Outlier identification in high dimensions, *Computational Statistics & Data Analysis*, Vol. 52 1694–1711.

P. Filzmoser & V. Todorov (2012), Robust tools for the imperfect world, To appear.

See Also

OutlierSign2, OutlierSign1, Outlier

Examples

```
data(hemophilia)
obj <- OutlierSign2(gr~.,data=hemophilia)
obj

getDistance(obj)            # returns an array of distances
getClassLabels(obj, 1)      # returns an array of indices for a given class
getCutoff(obj)           # returns an array of cutoff values (for each class, usually equal)
getFlag(obj)                # returns an 0/1 array of flags
plot(obj, class=2)          # standard plot function
```

OutlierSign2-class *Class "OutlierSign2" - Outlier identification in high dimensions using the SIGN2 algorithm*

Description

Fast algorithm for identifying multivariate outliers in high-dimensional and/or large datasets, using spatial signs, see Filzmoser, Maronna, and Werner (CSDA, 2007). The computation of the distances is based on principal components.

Objects from the Class

Objects can be created by calls of the form `new("OutlierSign2", ...)` but the usual way of creating `OutlierSign2` objects is a call to the function `OutlierSign2()` which serves as a constructor.

Slots

`covobj`: A list containing intermediate results of the SIGN2 algorithm for each class

`call`, `counts`, `grp`, `wt`, `flag`, `method`, `singularity`: from the "`Outlier`" class.

Extends

Class "`Outlier`", directly.

Methods

getCutoff Return the cutoff value used to identify outliers

getDistance Return a vector containing the computed distances

Author(s)

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

References

P. Filzmoser, R. Maronna and M. Werner (2008), Outlier identification in high dimensions, *Computational Statistics & Data Analysis*, Vol. 52 1694–1711.

P. Filzmoser & V. Todorov (2012), Robust tools for the imperfect world, To appear.

See Also

`OutlierSign2`, `OutlierSign1`, `Outlier`

Examples

```
showClass("OutlierSign2")
```

`PredictSimca-class` *Class* "`PredictSimca`" - *prediction of* "`Simca`" *objects*

Description

The prediction of a "Simca" object

Objects from the Class

Objects can be created by calls of the form `new("PredictSimca", ...)` but most often by invoking `predict()` on a "`Simca`" object. They contain values meant for printing by `show()`

Slots

`classification`: Object of class `"factor"` ~~

`odsc`: A `"matrix"` containing the standartized orthogonal distances for each group

`sdsc`: A `"matrix"` containing the standartized score distances for each group

`ct`: re-classification table of the training sample

Methods

show `signature(object = "PredictSimca")`: 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

Simca-class

Examples

```
showClass("PredictSimca")
```

`RSimca` *Robust classification in high dimensions based on the SIMCA method*

Description

RSimca performs a robust version of the SIMCA method. This method classifies a data matrix x with a known group structure. To reduce the dimension on each group a robust PCA analysis is performed. Afterwards a classification rule is developped to determine the assignment of new observations.

Usage

```
RSimca(x, ...)
## Default S3 method:
RSimca(x, grouping, prior=proportions, k, kmax = ncol(x),
    control="hubert", alpha, tol = 1.0e-4, trace=FALSE, ...)
## S3 method for class 'formula'
RSimca(formula, data = NULL, ..., subset, na.action)
```

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
control	a control object (S4) for specifying one of the available PCA estimation methods and 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, hubert, locantore, grid, proj. If 'auto' is specified or the argument is missing, the function will select the estimator (see below for details)
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.
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.
trace	whether to print intermediate results. Default is trace = FALSE
...	arguments passed to or from other methods.

Details

RSimca, serving as a constructor for objects of class RSimca-class is a generic function with "formula" and "default" methods.

SIMCA is a two phase procedure consisting of PCA performed on each group separately for dimension reduction followed by classification rules built in the lower dimensional space (note that the dimension in each group can be different). Instead of classical PCA robust alternatives will be used. Any of the robust PCA methods available in package Pca-class can be used through the argument control. In original SIMCA new observations are classified by means of their deviations from the

different PCA models. Here the classification rules will be obtained using two popular distances arising from PCA - orthogonal distances (OD) and score distances (SD). For the definition of these distances, the definition of the cutoff values and the standartization of the distances see Vanden Branden K, Hubert M (2005) and Todorov and Filzmoser (2009).

Value

An S4 object of class RSimca-class which is a subclass of of the virtual class Simca-class.

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Vanden Branden K, Hubert M (2005) Robust classification in high dimensions based on the SIMCA method. Chemometrics and Intellegent Laboratory Systems 79:10–21

Examples

```
data(pottery)
dim(pottery)        # 27 observations in 2 classes, 6 variables
head(pottery)

## Build the SIMCA model. Use RSimca for a robust version
rs <- RSimca(origin~., data=pottery)
rs
summary(rs)


## generate a sample from the pottery data set -
##  this will be the "new" data to be predicted
smpl <- sample(1:nrow(pottery), 5)
test <- pottery[smpl, -7]        # extract the test sample. Remove the last (grouping) variable
print(test)


## predict new data
pr <- predict(rs, newdata=test)

pr@classification
```

RSimca-class *Class* "RSimca" - robust classification in high dimensions based on the SIMCA method

---

Description

The class RSimca represents robust version of the SIMCA algorithm for classification in high dimensions. The objects of class RSImca contain the results of the robust SIMCA method.

Objects from the Class

Objects can be created by calls of the form `new("RSImca", ...)` but the usual way of creating `RSimca` objects is a call to the function `RSimca()` 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

`pcaobj`: A list of Pca objects - one for each group

`k`: Object of class `"numeric"` number of (choosen) principal components

`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

`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 `"Simca"`, directly.

Methods

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

Author(s)

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

References

Vanden Branden K, Hubert M (2005) Robust classification in high dimensions based on the SIMCA method. Chemometrics and Intellegent Laboratory Systems 79:10–21

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("RSimca")
```

Simca-class *Class "Simca" - virtual base class for all classic and robust SIMCA classes representing classification in high dimensions based on the SIMCA method*

Description

The class Simca searves as a base class for deriving all other classes representing the results of the classical and robust SIMCA 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

pcaobj: A list of Pca objects - one for each group

k: Object of class "numeric" number of (choosen) principal components

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

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.

Methods

predict signature(object = "Simca"): 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 = "Simca"): prints the results

summary signature(object = "Simca"): prints summary information

Author(s)

Valentin Todorov <valentin.todorov@chello.at>

References

Vanden Branden K, Hubert M (2005) Robust classification in high dimensions based on the SIMCA method. Chemometrics and Intellegent Laboratory Systems 79:10–21

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("Simca")
```

`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 Classi

cation 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 <- CSimca(F~., data=soil, k=c(3,3,1))
cc
pr <- predict(cc, method=2)
tt <- rrcov::mtxconfusion(cc@grp, pr@classification, printit=TRUE)
```

SPcaGrid — *Sparse Robust Principal Components based on Projection Pursuit (PP): GRID search Algorithm*

Description

Computes an approximation of the PP-estimators for sparse and robust PCA using the grid search algorithm in the plane.

Usage

```
    SPcaGrid(x, ...)
    ## Default S3 method:
SPcaGrid(x, k = 0, kmax = ncol(x), method = c ("mad", "sd", "qn", "Qn"),
    lambda = 1, scale=FALSE, na.action = na.fail, trace=FALSE, ...)
    ## S3 method for class 'formula'
SPcaGrid(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=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.
method	the scale estimator used to detect the direction with the largest variance. Possible values are "sd", "mad" and "Qn". "mad" is the default value.
lambda	the sparseness constraint's strength(sPCAgrid only). A single value for all components, or a vector of length k with different values for each component can be specified. See opt.TPO for the choice of this argument.
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
trace	whether to print intermediate results. Default is trace = FALSE

Details

SPcaGrid, serving as a constructor for objects of class SPcaGrid-class is a generic function with "formula" and "default" methods. For details see sPCAgrid and the relevant references.

Value

An S4 object of class SPcaGrid-class which is a subclass of PcaGrid-class which in turn 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*, Vol. 87, pp. 218-225.

C. Croux, P. Filzmoser, H. Fritz (2011). Robust Sparse Principal Component Analysis Based on Projection-Pursuit, To appear.

Examples

```
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")

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


## MADN vary between 0 (for variable 9) and 34. Therefore exclude
##  variable 9 and divide the remaining variables by their MADNs.
bus1 <- bus[, -c(9)]
p <- ncol(bus1)

madbus <- apply(bus1, 2, mad)
bus2 <- sweep(bus1, 2, madbus, "/", check.margin = FALSE)

xsd <- apply(bus1, 2, sd)
bus.sd <- sweep(bus1, 2, xsd, "/", check.margin = FALSE)

xqn <- apply(bus1, 2, Qn)
bus.qn <- sweep(bus1, 2, xqn, "/", check.margin = FALSE)

## Not run:
spc <- SPcaGrid(bus2, lambda=0, method="sd", k=p, kmax=p)
rspc <- SPcaGrid(bus2, lambda=0, method="Qn", k=p, kmax=p)
summary(spc)
summary(rspc)
screeplot(spc, type="line", main="Classical PCA", sub="PC", cex.main=2)
screeplot(rspc, type="line", main="Robust PCA", sub="PC", cex.main=2)

##  find lambda

K <- 4
lambda.sd <- 1.64
    to.sd <- .tradeoff(bus2, k=K, lambda.max=2.5, lambda.n=100, method="sd")
    plot(to.sd, type="b", xlab="lambda", ylab="Explained Variance (percent)")
    abline(v=lambda.sd, lty="dotted")

spc.sd.p <- SPcaGrid(bus2, lambda=lambda.sd, method="sd", k=p)
.CPEV(spc.sd.p, k=K)
spc.sd <- SPcaGrid(bus2, lambda=lambda.sd, method="sd", k=K)
getLoadings(spc.sd)[,1:K]
plot(spc.sd)

lambda.qn <- 2.06
    to.qn <- .tradeoff(bus2, k=K, lambda.max=2.5, lambda.n=100, method="Qn")
```

```
    plot(to.qn, type="b", xlab="lambda", ylab="Explained Variance (percent)")
    abline(v=lambda.qn, lty="dotted")

spc.qn.p <- SPcaGrid(bus2, lambda=lambda.qn, method="Qn", k=p)
.CPEV(spc.qn.p, k=K)
spc.qn <- SPcaGrid(bus2, lambda=lambda.qn, method="Qn", k=K)
getLoadings(spc.qn)[,1:K]
plot(spc.qn)

## End(Not run)

## DD-plots
##
## Not run:
## Not run:
usr <- par(mfrow=c(2,2))
plot(SPcaGrid(bus2, lambda=0, method="sd", k=4), id.n.sd=0, main="Standard PCA")
plot(SPcaGrid(bus2, lambda=0, method="Qn", k=4), id.n.sd=0, ylim=c(0,20))

plot(SPcaGrid(bus2, lambda=1.64, method="sd", k=4), id.n.sd=0, main="Stdandard sparse PCA")
plot(SPcaGrid(bus2, lambda=3.07, method="Qn", k=4), id.n.sd=0, main="Robust sparse PCA")

par(usr)
## End (Not run)

## End(Not run)
```

`SPcaGrid-class` *Class* `SPcaGrid` - *Sparse Robust PCA using PP - GRID search Algorithm*

Description

Holds the results of an approximation of the PP-estimators for sparse and robust PCA using the grid search algorithm in the plane.

Objects from the Class

Objects can be created by calls of the form `new("SPcaGrid", ...)` but the usual way of creating `SPcaGrid` objects is a call to the function `SPcaGrid()` 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`" class.

Extends

Class "`PcaGrid`", directly. Class "`PcaRobust`", by class "PcaGrid", distance 2. Class "`Pca`", by class "PcaGrid", distance 3.

Methods

getQuan `signature(obj = "SPcaGrid")`: ...

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

SPcaGrid, PcaGrid-class, PcaRobust-class, Pca-class, PcaClassic, PcaClassic-class

Examples

```
showClass("SPcaGrid")
```

SummarySimca-class *Class* "SummarySimca" - *summary of* "Simca" *objects*

Description

Contains summary information about a Simca object - classification in high dimensions based on the SIMCA method

Objects from the Class

Objects can be created by calls of the form `new("SummarySimca", ...)`, but most often by invoking `summary()` on an "Simca" object. They contain values meant for printing by `show()`.

Slots

simcaobj: Object of class "Simca"

Methods

show `signature(object = "SummarySimca")`: 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

Simca-class

Examples

```
showClass("SummarySimca")
```

Index