Package ‘amap’

February 19, 2015

Version 0.8-14
Date 2014-12-17
Title Another Multidimensional Analysis Package
Author Antoine Lucas
Maintainer Antoine Lucas <antoinelucas@gmail.com>
Depends R (>= 2.10.0)
Suggests Biobase
Description Tools for Clustering and Principal Component Analysis
   (With robust methods, and parallelized functions).
License GPL
URL http://mulcyber.toulouse.inra.fr/projects/amap/
Repository CRAN
Date/Publication 2014-12-17 23:26:42
NeedsCompilation yes

R topics documented:

acp ................................................................. 2
acpgen .......................................................... 3
acprob ........................................................... 5
afc ................................................................. 7
burt ............................................................... 7
diss ............................................................... 8
Dist ............................................................... 9
hcluster ........................................................ 12
Kmeans .......................................................... 15
lubisch ......................................................... 16
plot ............................................................. 16
pop ............................................................. 18
VarRob ........................................................ 19

Index 21
Principal component analysis

Description
Principal component analysis

Usage
acp(x, center = TRUE, reduce = TRUE, wI = rep(1, nrow(x)), wV = rep(1, ncol(x)))
pca(x, center = TRUE, reduce = TRUE, wI = rep(1, nrow(x)), wV = rep(1, ncol(x)))
## S3 method for class 'acp'
print(x, ...)

Arguments
x Matrix / data frame
center a logical value indicating whether we center data
reduce a logical value indicating whether we "reduce" data i.e. divide each column by standard deviation
wI, wV weigth vector for individuals / variables
... arguments to be passed to or from other methods.

Details
This function offer a variant of princomp and prcomp functions, with a slightly different graphic representation (see plot.acp).

Value
An object of class acp. The object is a list with components:
sdev the standard deviations of the principal components.
loadings the matrix of variable loadings (i.e., a matrix whose columns contain the eigen-vectors). This is of class "loadings": see loadings for its print method.
scores if scores = TRUE, the scores of the supplied data on the principal components.
eig Eigen values

Author(s)

References
A. Carlier Analyse des donn\'ees Multidimensionnelles http://www.lsp.ups-tlse.fr/Carlier/enseignement.html
acpgen

See Also

plot.acp, acpgen, princomp

Examples

data(lubisch)
lubisch <- lubisch[, -c(1, 8)]
p <- acp(lubisch)
plot(p)

Description

Generalised principal component analysis

Usage

acpgen(x, h1, h2, center = TRUE, reduce = TRUE, kernel = "gaussien")
K(u, kernel = "gaussien")
W(x, h, D = NULL, kernel = "gaussien")

Arguments

x Matrix or data frame
h Scalar: bandwidth of the Kernel
h1 Scalar: bandwidth of the Kernel for W
h2 Scalar: bandwidth of the Kernel for U
kernel The kernel used. This must be one of "gaussien", "quartic", "triweight", "epanechikov", "cosinus" or "uniform"
center A logical value indicating whether we center data
reduce A logical value indicating whether we "reduce" data i.e. divide each column by standard deviation
D A product scalar matrix / une matrice de produit scalaire
u A vector

Details

acpgen compute generalised pca. i.e. spectral analysis of \( U_n W_n^{-1} \) and project \( X_i \) with \( W_n^{-1} \) on the principal vector sub-spaces.

\( X_i \) a column vector of \( p \) variables of individu \( i \) (input data)
\[ W_n = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} K(||X_i - X_j||_{V_n^{-1}/h})(X_i - X_j)(X_i - X_j)' \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} K(||X_i - X_j||_{V_n^{-1}/h})}{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} K(||X_i - X_j||_{V_n^{-1}/h})} \]

with \( V_n \) variance estimation;

\[ U_n^{-1} = S_n^{-1} - 1/h V_n^{-1} \]

\[ S_n = \frac{\sum_{i=1}^{n} K(||X_i||_{V_n^{-1}/h})(X_i - \mu_n)(X_i - \mu_n)' \sum_{i=1}^{n} K(||X_i||_{V_n^{-1}/h})}{\sum_{i=1}^{n} K(||X_i||_{V_n^{-1}/h})} \]

with \( \mu_n \) estimator of the mean.

K compute kernel, i.e.

gaussian:
\[ \frac{1}{\sqrt{2\pi}} e^{-u^2/2} \]

quartic:
\[ \frac{15}{16} (1 - u^2)^2 I_{|u|\leq 1} \]

triweight:
\[ \frac{35}{32} (1 - u^2)^3 I_{|u|\leq 1} \]

epanechnikov:
\[ \frac{3}{4} (1 - u^2) I_{|u|\leq 1} \]

cosinus:
\[ \frac{\pi}{4} \cos\left(\frac{\pi}{2} u\right) I_{|u|\leq 1} \]

Value

An object of class \texttt{acp} The object is a list with components:

\texttt{sdev} the standard deviations of the principal components.
\texttt{loadings} the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). This is of class "loadings": see \texttt{loadings} for its print method.
\texttt{scores} if \texttt{scores = TRUE}, the scores of the supplied data on the principal components.
\texttt{eig} Eigen values

Author(s)
Antoine Lucas, \url{http://mulcyber.toulouse.inra.fr/projects/amap/}
acprob

References


See Also

acp acprob princomp

Examples

data(lubisch)
lubisch <- lubisch[-c(1,8)]
p <- acpgen(lubisch,h1=1,h2=1/sqrt(2))
plot(p,main='ACP robuste des individus')

# See difference with acp

p <- princomp(lubisch)
class(p)<- "acp"

acprob

Robust principal component analysis

Description

Robust principal component analysis

Usage

acprob(x,h,center=TRUE,reduce=TRUE,kernel="gaussien")

Arguments

x Matrix / data frame
h Scalar: bandwidth of the Kernel
kernel The kernel used. This must be one of "gaussien", "quartic", "triweight", "epanechikov", "cosinus" or "uniform"
center A logical value indicating whether we center data
reduce A logical value indicating whether we "reduce" data i.e. divide each column by standard deviation

Details

acpgen compute robust pca. i.e. spectral analysis of a robust variance instead of usual variance.
Robust variance: see varrob

Value

An object of class acp The object is a list with components:

sdev the standard deviations of the principal components.
loadings the matrix of variable loadings (i.e., a matrix whose columns contain the eigenvectors). This is of class "loadings": see loadings for its print method.
scores if scores = TRUE, the scores of the supplied data on the principal components.
eig Eigen values

Author(s)


References


See Also

princomp acpgen
**afc**

*Correspondance factorial analysis.*

**Description**

Compute an acp on a contingency table tacking into account weight of rows and columns

**Usage**

```r
afc(x)
```

**Arguments**

- `x` A contingency table, or a result of function `burt` or `matlogic`

**Author(s)**


**Examples**

```r
## Not run:
color <- as.factor(c('blue','red','red','blue','red'))
size <- as.factor(c('large','large','small','medium','large'))
x <- data.frame(color,size)

afc.1 <- afc(burt(x))
afc.2 <- afc(matlogic(x))

plotAll(afc.1)
plotAll(afc.2)

## End(Not run)
```

---

**burt**

*Compute burt table from a factor dataframe.*

**Description**

matlogic returns for all variables a matrix of logical values for each levels. burt is defined as t(matlogic).matlogic

**Usage**

```r
burt(x)
matlogic(x)
```
**Arguments**

\( x \)  
A dataframe that contains only factors

**Author(s)**


**Examples**

color <- as.factor(c('blue','red','red','blue','red'))  
size <- as.factor(c('large','large','small','medium','large'))  
x <- data.frame(color, size)

```r
matlogic(x)
## color.blue color.red size.large size.medium size.small
## #1 1 0 1 0 0
## #2 0 1 1 0 0
## #3 0 1 0 0 1
## #4 1 0 0 1 0
## #5 0 1 1 0 0
```

```r
burt(x)
## color.blue color.red size.large size.medium size.small
## # color.blue 2 0 1 1 0
## # color.red 0 3 2 0 1
## # size.large 1 2 3 0 0
## # size.medium 1 0 0 1 0
## # size.small 0 1 0 0 1
```

**diss**  
*Compute a dissimilarity matrix*

**Description**

Compute a dissimilarity matrix from a data set (containing only factors).

**Usage**

diss(x, w=rep(1,ncol(x)))

**Arguments**

\( x \)  
A matrix or data frame containing only factors.

\( w \)  
A vector of weight, by default each variable has got same weight.
Details

Case of N individuals described by P categorical variables: each element (i,j) of the signed similarities array is computed by summation over the P variables of the contributions of each variable, multiplied by the weight of the variable. The contribution of a given categorical variable is +1 if the individual i and j are in the same class, and is -1 if they are not.

Value

A dissimilarity matrix.

Author(s)

Antoine Lucas

See Also

Dist, pop

Examples

```r
data <- matrix(c(1,1,1,1,1,
                 1,2,1,2,1,
                 2,3,2,3,2,
                 2,4,3,3,2,
                 1,2,4,2,1,
                 2,3,2,3,1), ncol=5, byrow=TRUE)

diss(data)

## With weights
diss(data, w=c(1,1,2,2,3))
```

Description

This function computes and returns the distance matrix computed by using the specified distance measure to compute the distances between the rows of a data matrix.

Usage

```r
Dist(x, method = "euclidean", nbproc = 2, diag = FALSE, upper = FALSE)
```
Arguments

x  numeric matrix or (data frame) or an object of class "exprSet". Distances between the rows of x will be computed.

method  the distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary", "pearson", "abspearson", "correlation", "abscorrelation", "spearman" or "kendall". Any unambiguous substring can be given.

nbproc  integer, Number of subprocess for parallelization

diag  logical value indicating whether the diagonal of the distance matrix should be printed by printNdist.

upper  logical value indicating whether the upper triangle of the distance matrix should be printed by printNdist.

Details

Available distance measures are (written for two vectors x and y):

euclidean: Usual square distance between the two vectors (2 norm).

maximum: Maximum distance between two components of x and y (supremum norm)

manhattan: Absolute distance between the two vectors (1 norm).

canberra: ∑(|x_i - y_i|/|x_i + y_i|). Terms with zero numerator and denominator are omitted from the sum and treated as if the values were missing.

binary: (aka asymmetric binary): The vectors are regarded as binary bits, so non-zero elements are ‘on’ and zero elements are ‘off’. The distance is the proportion of bits in which only one is on.

pearson: Also named "not centered Pearson" 1 − \frac{\sum x_i y_i}{\sqrt{\sum x_i^2 \sum y_i^2}}.

abspearson: Absolute Pearson 1 − \frac{\sum |x_i y_i|}{\sqrt{\sum x_i^2 \sum y_i^2}}.

correlation: Also named "Centered Pearson" 1 − \text{corr}(x, y).

abscorrelation: Absolute correlation 1 − |\text{corr}(x, y)| with\text{corr}(x, y) = \frac{\sum x_i y_i - \frac{1}{n} \sum x_i \sum y_i}{\sqrt{\left(\sum x_i^2 - \frac{1}{n} (\sum x_i)^2\right) \left(\sum y_i^2 - \frac{1}{n} (\sum y_i)^2\right)}}.

spearman: Compute a distance based on rank. \sum(d_i^2) where d_i is the difference in rank between x_i and y_i.

Dist(x, method="spearman")[i,j] = cor.test(x[i,], x[j,], method="spearman")$statistic

kendall: Compute a distance based on rank. \sum_{i,j} K_{i,j}(x, y) with K_{i,j}(x, y) = 0 if x_i, x_j in same order as y_i, y_j, 1 if not.

Missing values are allowed, and are excluded from all computations involving the rows within which they occur. If some columns are excluded in calculating a Euclidean, Manhattan or Canberra distance, the sum is scaled up proportionally to the number of columns used. If all pairs are excluded when calculating a particular distance, the value is NA.

The functions as.matrix.dist() and as.dist() can be used for conversion between objects of class "dist" and conventional distance matrices and vice versa.
Value

An object of class "dist".

The lower triangle of the distance matrix stored by columns in a vector, say do. If \( n \) is the number of observations, i.e., \( n \leftarrow \text{attr}(d, \text{"Size"}) \), then for \( i < j <= n \), the dissimilarity between (row) \( i \) and \( j \) is \( d(n(i-1) - i*(i-1)/2 + j-i) \). The length of the vector is \( n * (n - 1)/2 \), i.e., of order \( n^2 \).

The object has the following attributes (besides "class" equal to "dist"):

- **Size**: integer, the number of observations in the dataset.
- **Labels**: optionally, contains the labels, if any, of the observations of the dataset.
- **Diag, Upper**: logicsls corresponding to the arguments diag and upper above, specifying how the object should be printed.
- **call**: optionally, the call used to create the object.
- **methods**: optionally, the distance method used; resulting form dist(), the (match.arg()ed) method argument.

Note

Multi-thread (parallelisation) is disable on Windows.

References


See Also

daisy in the 'cluster' package with more possibilities in the case of mixed (continuous / categorical) variables. dist hcluster.

Examples

```r
x <- matrix(rnorm(100), nrow=5)
Dist(x)
Dist(x, diag = TRUE)
Dist(x, upper = TRUE)

## compute dist with 8 threads
Dist(x,nbproc=8)

Dist(x,method="abscorrelation")
Dist(x,method="kendall")
```
hcluster  

Hierarchical Clustering

Description
Hierarchical cluster analysis.

Usage
hcluster(x, method = "euclidean", diag = FALSE, upper = FALSE, link = "complete", members = NULL, nbproc = 2, doubleprecision = TRUE)

Arguments
x  A numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns). Or an object of class "exprSet".
method  the distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary", "pearson", "abspearson", "correlation", "abscorrelation", "spearman" or "kendall". Any unambiguous substring can be given.
diag  logical value indicating whether the diagonal of the distance matrix should be printed by print.dist.
upper  logical value indicating whether the upper triangle of the distance matrix should be printed by print.dist.
link  the agglomeration method to be used. This should be (an unambiguous abbreviation of) one of "ward", "single", "complete", "average", "mcquitty", "median" or "centroid","centroid2".
members  NULL or a vector with length size of d.
nbproc  integer, number of subprocess for parallelization [Linux & Mac only]
doubleprecision  True: use of double precision for distance matrix computation; False: use simple precision

Details
This function is a mix of function hclust and function dist. hcluster(x, method = "euclidean",link = "complete") = hclust(dist(x, method = "euclidean"),method = "complete") It use twice less memory, as it doesn't store distance matrix.

For more details, see documentation of hclust and Dist.
Value

An object of class `hclust` which describes the tree produced by the clustering process. The object is a list with components:

- **merge**: an \( n - 1 \) by 2 matrix. Row \( i \) of `merge` describes the merging of clusters at step \( i \) of the clustering. If an element \( j \) in the row is negative, then observation \( -j \) was merged at this stage. If \( j \) is positive then the merge was with the cluster formed at the (earlier) stage \( j \) of the algorithm. Thus negative entries in `merge` indicate agglomerations of singletons, and positive entries indicate agglomerations of non-singletons.

- **height**: a set of \( n - 1 \) non-decreasing real values. The clustering `height`: that is, the value of the criterion associated with the clustering method for the particular agglomeration.

- **order**: a vector giving the permutation of the original observations suitable for plotting, in the sense that a cluster plot using this ordering and matrix `merge` will not have crossings of the branches.

- **labels**: labels for each of the objects being clustered.

- **call**: the call which produced the result.

- **method**: the cluster method that has been used.

- **dist.method**: the distance that has been used to create \( d \) (only returned if the distance object has a "method" attribute).

There is a `print` and a `plot` method for `hclust` objects. The `plclust()` function is basically the same as the plot method, `plot.hclust`, primarily for back compatibility with S-plus. Its extra arguments are not yet implemented.

Note

Multi-thread (parallelisation) is disable on Windows.

Author(s)

The `hcluster` function is based on C code adapted from Cran Fortran routine by Antoine Lucas [http://mulcyber.toulouse.inra.fr/projects/amap/](http://mulcyber.toulouse.inra.fr/projects/amap/).

References


See Also

`Dist`, `hclust`, `kmeans`.
Examples

data(USArrests)
hc <- hclust(USArrests, link = "ave")
plot(hc)
plot(hc, hang = -1)

## Do the same with centroid clustering and squared Euclidean distance,
## cut the tree into ten clusters and reconstruct the upper part of the
## tree from the cluster centers.
hc <- hclust(dist(USArrests)^2, "cen")
memb <- cutree(hc, k = 10)
cent <- NULL
for(k in 1:10){
  cent <- rbind(cent, colMeans(USArrests[memb == k, , drop = FALSE]))
}
hc1 <- hclust(dist(cent)^2, method = "cen", members = table(memb))
opar <- par(mfrow = c(1, 2))
plot(hc, labels = FALSE, hang = -1, main = "Original Tree")
plot(hc1, labels = FALSE, hang = -1, main = "Re-start from 10 clusters")
par(opar)

## other combinaison are possible

hc <- hcluster(USArrests, method = "euc", link = "ward", nbproc= 1,
  doubleprecision = TRUE)
hc <- hcluster(USArrests, method = "max", link = "single", nbproc= 2,
  doubleprecision = TRUE)
hc <- hcluster(USArrests, method = "man", link = "complete", nbproc= 1,
  doubleprecision = TRUE)
hc <- hcluster(USArrests, method = "can", link = "average", nbproc= 2,
  doubleprecision = TRUE)
hc <- hcluster(USArrests, method = "bin", link = "mcquitty", nbproc= 1,
  doubleprecision = FALSE)
hc <- hcluster(USArrests, method = "pea", link = "median", nbproc= 2,
  doubleprecision = FALSE)
hc <- hcluster(USArrests, method = "abspea", link = "median", nbproc= 2,
  doubleprecision = FALSE)
hc <- hcluster(USArrests, method = "cor", link = "centroid", nbproc= 1,
  doubleprecision = FALSE)
hc <- hcluster(USArrests, method = "abscor", link = "centroid", nbproc= 1,
  doubleprecision = FALSE)
hc <- hcluster(USArrests, method = "spe", link = "complete", nbproc= 2,
  doubleprecision = FALSE)
hc <- hcluster(USArrests, method = "ken", link = "complete", nbproc= 2,
  doubleprecision = FALSE)
**Kmeans**

**K-Means Clustering**

**Description**

Perform k-means clustering on a data matrix.

**Usage**

\[
\text{Kmeans}(x, \text{centers}, \text{iter.max} = 10, \text{nstart} = 1, \\
\text{method} = "euclidean")
\]

**Arguments**

- **x**: A numeric matrix of data, or an object that can be coerced to such a matrix (such as a numeric vector or a data frame with all numeric columns). Or an object of class "exprSet".
- **centers**: Either the number of clusters or a set of initial cluster centers. If the first, a random set of rows in \(x\) are chosen as the initial centers.
- **iter.max**: The maximum number of iterations allowed.
- **nstart**: If \(\text{centers}\) is a number, how many random sets should be chosen?
- **method**: The distance measure to be used. This must be one of "euclidean", "maximum", "manhattan", "canberra", "binary", "pearson", "abspearson", "abscorrelation", "correlation", "spearman" or "kendall". Any unambiguous substring can be given.

**Details**

The data given by \(x\) is clustered by the k-means algorithm. When this terminates, all cluster centres are at the mean of their Voronoi sets (the set of data points which are nearest to the cluster centre).

The algorithm of Lloyd–Forgy is used; method="euclidean" should return same result as with function \text{kmeans}.

**Value**

A list with components:

- **cluster**: A vector of integers indicating the cluster to which each point is allocated.
- **centers**: A matrix of cluster centres.
- **withinss**: The within-cluster sum of square distances for each cluster.
- **size**: The number of points in each cluster.

**Note**

An objective: to allow NA values.
See Also
   hcluster, kmeans.

Examples

```r
# a 2-dimensional example
x <- rbind(matrix(rnorm(100, sd = 0.3), ncol = 2),
           matrix(rnorm(100, mean = 1, sd = 0.3), ncol = 2))
colnames(x) <- c("x", "y")
(cl <- Kmeans(x, 2))
plot(x, col = cl$cluster)
points(cl$centers, col = 1:2, pch = 8, cex=2)

# random starts do help here with too many clusters
(cl <- Kmeans(x, 5, nstart = 25))
plot(x, col = cl$cluster)
points(cl$centers, col = 1:5, pch = 8)

Kmeans(x, 5, nstart = 25, method="abscorrelation")
```

---

### lubisch

**Dataset Lubischew**

Description

Lubischew data (1962): 74 insects, 6 morphologic size. 3 supposed classes

Usage

data(lubisch)

---

### plot

**Graphics for Principal component Analysis**

Description

Graphics for Principal component Analysis
**Usage**

```r
## S3 method for class 'acp'
plot(x, i = 1, j = 2, text = TRUE, label = 'Composants', col = 'darkblue',
     main = 'Individuals PCA', variables = TRUE, individual.label = NULL,...)
## S3 method for class 'acp'
biplot(x, i = 1, j = 2, label = 'Composants', col = 'darkblue', length = 0.1,
       main = 'Variables PCA', circle = TRUE,...)
plot2(x, pourcent = FALSE, eigen = TRUE, label = 'Comp.', col = 'lightgrey',
      main = 'Scree Graph', ylab = 'Eigen Values')
plotAll(x)
```

**Arguments**

- `x` Result of acp or princomp
- `i` X axis
- `j` Y axis
- `text` a logical value indicating whether we use text or points for plot
- `pourcent` a logical value indicating whether we use percentage of values
- `eigen` a logical value indicating whether we use eigen values or standard deviation
- `label` label for X and Y axis
- `individual.label` labels naming individuals
- `col` Color of plot
- `main` Title of graphic
- `ylab` Y label
- `length` length of arrows
- `variables.circle` a logical value indicating whether we display circle or variables
- `...` cex, pch, and other options; see points.

**Value**

Graphics:
- `plot.acp` PCA for lines (individuals)
- `plot.acp` PCA for columns (variables)
- `plot2` Eigen values diagram (Scree Graph)
- `plotAll` Plot both 3 graphs

**Author(s)**


**See Also**

`acpgen, acprob, princomp`
Examples

```r
data(lubisch)
lubisch <- lubisch[, -c(1, 8)]
p <- acp(lubisch)
plotAll(p)
```

---

**pop**

*Optimal Partition (classification).*

Description

Classification: Computing an Optimal Partition from Weighted Categorical Variables or from an Array of Signed Similarities.

Usage

```r
pop(x, fmbvr = TRUE, triabs = TRUE, allsol = TRUE)
```

Arguments

- **x**: A dissimilarity matrix
- **fmbvr**: Logical, TRUE: look for the exact solution
- **triabs**: Logical, TRUE: try to init with absolute values
- **allsol**: Logical, TRUE all solutions, FALSE only one solution

Author(s)

Michel Petitjean, [http://petitjeanmichel.free.fr/itoweb.petitjean.class.html](http://petitjeanmichel.free.fr/itoweb.petitjean.class.html)


References

Theory is explained at [http://petitjeanmichel.free.fr/itoweb.petitjean.class.html](http://petitjeanmichel.free.fr/itoweb.petitjean.class.html)


Examples

```r
## pop from a data matrix
data <- matrix(c(1,1,1,1,1,2,1,2,1,2,3,2,3,2,2,4,3,2,1,2,4,2,1,2,3,2,3,2,2,3,2,3,2,2,3,2,3,2,2,3,2),nrow=5,ncol=5,byrow=TRUE)

pop(diss(data))

## pop from a dissimilarity matrix

d <- 2 * matrix(c(9,8,5,7,7,2,8,9,2,5,1,7,5,2,9,8,7,1,7,5,8,9,3,2,7,1,7,3,9,6,2,7,1,2,6,9),ncol=6,byrow=TRUE) - 9

pop(d)

## Not run:
d <- 2 * matrix(c(57,15,11,32,1,34,4,6,17,7,15,57,27,35,27,27,20,24,30,15,11,27,57,25,25,20,34,25,17,15,32,35,25,57,22,44,13,22,30,11,1,27,25,22,57,21,28,43,20,13,34,27,20,44,21,57,18,27,21,8,4,20,34,13,28,18,57,31,28,13,6,24,25,22,43,27,31,57,30,15,17,30,17,30,20,21,28,30,57,12,7,15,15,11,13,8,13,15,12,57),ncol=10,byrow=TRUE) - 57

pop(d)

## End(Not run)
```

---

VarRob  Robust variance
Description

Compute a robust variance

Usage

varrob(x, h, D=NULL, kernel="gaussien")

Arguments

x Matrix / data frame
h Scalar: bandwidth of the Kernel
kernel The kernel used. This must be one of "'gaussien'"", "quartic", "triweight", "epanechikov", "cosinus" or "uniform"
D A product scalar matrix / une matrice de produit scalaire

Details

U compute robust variance. $U_n^{-1} = S_n^{-1} - 1/h V_n^{-1}$

$$S_n = \frac{\sum_{i=1}^{n} K(||X_i||V_n^{-1}/h)(X_i - \mu_n)(X_i - \mu_n)' \sum_{i=1}^{n} K(||X_i||V_n^{-1}/h)}$$

with $\mu_n$ estimator of the mean.

K compute a kernel.

Value

A matrix

Author(s)


References


See Also

acp princomp
Index

*Topic **cluster**
  diss, 8
  Dist, 9
  hcluster, 12
  Kmeans, 15

*Topic **datasets**
  lubisch, 16

*Topic **multivariate**
  acp, 2
  acpgen, 3
  acprob, 5
 afc, 7
  burt, 7
  diss, 8
  Dist, 9
  hcluster, 12
  Kmeans, 15
  plot, 16
  pop, 18
  VarRob, 19

  acp, 2, 5, 20
  acpgen, 3, 3, 6, 17
  acprob, 5, 5, 17
  afc, 7

  biplot.acp (plot), 16
  burt, 7

  call, 11

  daisy, 11
  diss, 8
  Dist, 9, 9, 13
  dist, 11

  hclust, 13
  hcluster, 11, 12, 16
  hclusterpar (hcluster), 12

  K (acpgen), 3

  Kmeans, 15
  kmeans, 13, 15, 16

  loadings, 2, 4, 6
  lubisch, 16

  match.arg, 11
  matlogic (burt), 7

  pca (acp), 2
  plot, 13, 16
  plot.acp, 2, 3
  plot.acp (plot), 16
  plot2 (plot), 16
  plotAll (plot), 16
  pop, 9, 18
  prcomp, 2
  princomp, 2, 3, 5, 6, 17, 20
  print, 13
  print.acp (acp), 2

  VarRob, 19
  varrob, 6
  varrob (VarRob), 19

  W (acpgen), 3