Package ‘fpc’

April 26, 2019

Title Flexible Procedures for Clustering

Version 2.1-11.2

Date 2018-01-05

Author Christian Hennig <c.hennig@ucl.ac.uk>

Depends R (>= 3.6.0)

Imports MASS, cluster, mclust, flexmix, prabclus, class, diptest,
mvtnorm, robustbase, kernlab, trimcluster, grDevices, graphics,
methods, stats, utils

Suggests tclust, pdfCluster

Description Various methods for clustering and cluster validation.
Fixed point clustering. Linear regression clustering. Clustering by
merging Gaussian mixture components. Symmetric
and asymmetric discriminant projections for visualisation of the
separation of groupings. Cluster validation statistics
for distance based clustering including corrected Rand index.
Cluster-wise cluster stability assessment. Methods for estimation of
the number of clusters: Calinski-Harabasz, Tibshirani and Walther's
prediction strength, Fang and Wang's bootstrap stability.
Gaussian/multinomial mixture fitting for mixed
continuous/categorical variables. Variable-wise statistics for cluster
interpretation. DBSCAN clustering. Interface functions for many
clustering methods implemented in R, including estimating the number of
clusters with kmeans, pam and clara. Modality diagnosis for Gaussian
mixtures. For an overview see package?fpc.

Maintainer ORPHANED

License GPL

URL http://www.homepages.ucl.ac.uk/~ucakche/

NeedsCompilation no

Repository CRAN

Date/Publication 2019-04-26 13:21:00 UTC

X-CRAN-Original-Maintainer Christian Hennig <c.hennig@ucl.ac.uk>
X-CRAN-Comment Orphaned and corrected on 2018-07-20 as check problems were not corrected despite reminders.

R topics documented:

fpc-package ........................................... 3
adcoord ................................................... 6
ancoord ................................................... 7
awcoord ................................................... 8
batcoord .................................................. 10
bhattacharyya.dist ..................................... 12
bhattacharyya.matrix ................................. 13
calinhara ............................................... 14
can ....................................................... 15
cat2bin .................................................. 16
cdbw ...................................................... 17
classifdist ............................................. 18
clucols ................................................... 20
clujaccard ............................................... 21
cluexpect ............................................... 21
cluster.stats .......................................... 22
cluster.varstats ....................................... 26
cluster.boot ........................................... 28
cmahal ................................................... 34
con.comp ............................................... 35
confusion ............................................... 36
cov.wml ................................................ 37
cvnn ..................................................... 38
cweight .................................................. 39
dbscan ................................................... 40
dipp.tantrum .......................................... 42
diptest.multi ......................................... 43
discrcoord ............................................. 44
discrete.recode ....................................... 45
discrproj ............................................... 47
distancefactor ........................................ 49
distcritmulti ......................................... 50
bridgeline ............................................. 52
dudahart2 ............................................. 53
extract.mixturepars .................................. 54
findrep .................................................. 55
fixmahal ............................................... 56
fixreg ................................................... 62
flexmixedruns ......................................... 68
fpclusters ............................................. 71
itnumber ............................................... 71
jittervar ............................................... 72
kmeansCBI ............................................. 73
Here is a list of the main functions in package fpc. Most other functions are auxiliary functions for these.
Clustering methods

- **dbscan** Computes DBSCAN density based clustering as introduced in Ester et al. (1996).
- **fixmahal** Mahalanobis Fixed Point Clustering, Hennig and Christlieb (2002), Hennig (2005).
- **fixreg** Regression Fixed Point Clustering, Hennig (2003).
- **flexmixedruns** This fits a latent class model to data with mixed type continuous/nominal variables. Actually it calls a method for flexmix.
- **mergenormals** Clustering by merging components of a Gaussian mixture, see Hennig (2010).
- **regmix** ML-fit of a mixture of linear regression models, see DeSarbo and Cron (1988).

Cluster validity indexes and estimation of the number of clusters

- **cluster.stats** This computes several cluster validity statistics from a clustering and a dissimilarity matrix including the Calinski-Harabasz index, the adjusted Rand index and other statistics explained in Gordon (1999) as well as several characterising measures such as average between cluster and within cluster dissimilarity and separation. See also calinhara, dudahart2 for specific indexes.
- **prediction.strength** Estimates the number of clusters by computing the prediction strength of a clustering of a dataset into different numbers of components for various clustering methods, see Tibshirani and Walther (2005). In fact, this is more flexible than what is in the original paper, because it can use point classification schemes that work better with clustering methods other than k-means.
- **nselectboot** Estimates the number of clusters by bootstrap stability selection, see Fang and Wang (2012). This is quite flexible regarding clustering methods and point classification schemes and also allows for dissimilarity data.

Cluster visualisation and validation

- **clucols** Sets of colours and symbols useful for cluster plotting.
- **clusterboot** Cluster-wise stability assessment of a clustering. Clusterings are performed on resampled data to see for every cluster of the original dataset how well this is reproduced. See Hennig (2007) for details.
- **cluster.varstats** Extracts variable-wise information for every cluster in order to help with cluster interpretation.
- **plotcluster** Visualisation of a clustering or grouping in data by various linear projection methods that optimise the separation between clusters, or between a single cluster and the rest of the data according to Hennig (2004) including classical methods such as discriminant coordinates. This calls the function discrproj, which is a bit more flexible but doesn’t produce a plot itself.
- **ridgeline.diagnosis** Plots and diagnostics for assessing modality of Gaussian mixtures, see Ray and Lindsay (2005).
- **weightplots** Plots to diagnose component separation in Gaussian mixtures, see Hennig (2010).
- **localshape** Local shape matrix, can be used for finding clusters in connection with function ics in package ICS, see Hennig’s discussion and rejoinder of Tyler et al. (2009).
Useful wrapper functions for clustering methods

**kmeansCBI** This and other "CBI"-functions (see the kmeansCBI-help page) are unified wrappers for various clustering methods in R that may be useful because they do in one step for what you normally may need to do a bit more in R (for example fitting a Gaussian mixture with noise component in package mclust).

**kmeansruns** This calls kmeans for the k-means clustering method and includes estimation of the number of clusters and finding an optimal solution from several starting points.

**pamk** This calls pam and clara for the partitioning around medoids clustering method (Kaufman and Rousseeuw, 1990) and includes two different ways of estimating the number of clusters.

**Author(s)**

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

**References**


Asymmetric discriminant coordinates as defined in Hennig (2003). Asymmetric discriminant projection means that there are two classes, one of which is treated as the homogeneous class (i.e., it should appear homogeneous and separated in the resulting projection) while the other may be heterogeneous. The principle is to maximize the ratio between the projection of a between classes separation matrix and the projection of the covariance matrix within the homogeneous class.

Usage

adcoord(xd, clvecd, clnum=1)

Arguments

xd the data matrix; a numerical object which can be coerced to a matrix.
clvecd integer vector of class numbers; length must equal nrow(xd).
clnum integer. Number of the homogeneous class.

Details

The square root of the homogeneous classes covariance matrix is inverted by use of tdecomp, which can be expected to give reasonable results for singular within-class covariance matrices.

Value

List with the following components

ev eigenvalues in descending order.
units columns are coordinates of projection basis vectors. New points x can be projected onto the projection basis vectors by x %*% units
proj projections of xd onto units.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


See Also

plotcluster for straight forward discriminant plots. discrproj for alternatives. rFace for generation of the example data used below.

Examples

```r
sset.seed(4634)
face <- rFace(600, dMoNo=2, dNoEy=0)
grface <- as.integer(attr(face,"grouping"))
adcf <- adcoord(face, grface==2)
adcf2 <- adcoord(face, grface==4)
plot(adcf$proj, col=1+(grface==2))
plot(adcf2$proj, col=1+(grface==4))
# ...done in one step by function plotcluster.
```

ancoord

Asymmetric neighborhood based discriminant coordinates

Description

Asymmetric neighborhood based discriminant coordinates as defined in Hennig (2003). Asymmetric discriminant projection means that there are two classes, one of which is treated as the homogeneous class (i.e., it should appear homogeneous and separated in the resulting projection) while the other may be heterogeneous. The principle is to maximize the ratio between the projection of a between classes covariance matrix, which is defined by averaging the between classes covariance matrices in the neighborhoods of the points of the homogeneous class and the projection of the covariance matrix within the homogeneous class.

Usage

```r
ancoord(xd, clvecd, clnum=1, nn=50, method="mcd", countmode=1000, ...)
```

Arguments

- `xd`: the data matrix; a numerical object which can be coerced to a matrix.
- `clvecd`: integer vector of class numbers; length must equal nrow(xd).
- `clnum`: integer. Number of the homogeneous class.
- `nn`: integer. Number of points which belong to the neighborhood of each point (including the point itself).
- `method`: one of "mve", "mcd" or "classical". Covariance matrix used within the homogeneous class. "mcd" and "mve" are robust covariance matrices as implemented in cov.rob. "classical" refers to the classical covariance matrix.
- `countmode`: optional positive integer. Every countmode algorithm runs ancoord shows a message.
- `...`: no effect
Details

The square root of the homogeneous classes covariance matrix is inverted by use of \texttt{tdecomp}, which can be expected to give reasonable results for singular within-class covariance matrices.

Value

List with the following components:

- \texttt{ev}: eigenvalues in descending order.
- \texttt{units}: columns are coordinates of projection basis vectors. New points \( x \) can be projected onto the projection basis vectors by \( x \% \% \texttt{units} \).
- \texttt{proj}: projections of \( x \) onto \texttt{units}.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> \url{http://www.homepages.ucl.ac.uk/~ucakche/}

References


See Also

\texttt{plotcluster} for straightforward discriminant plots. \texttt{discrproj} for alternatives. \texttt{rFace} for generation of the example data used below.

Examples

```
set.seed(4634)
face <- rFace(600,dMoNo=2,dNoEy=0)
grface <- as.integer(attr(face,"grouping"))
ancf2 <- ancoord(face,grface==4)
plot(ancf2$proj,col=1+(grface==4))
# ... done in one step by function plotcluster.
```

Description

Asymmetric weighted discriminant coordinates as defined in Hennig (2003). Asymmetric discriminant projection means that there are two classes, one of which is treated as the homogeneous class (i.e., it should appear homogeneous and separated in the resulting projection) while the other may be heterogeneous. The principle is to maximize the ratio between the projection of a between classes separation matrix and the projection of the covariance matrix within the homogeneous class. Points are weighted according to their (robust) Mahalanobis distance to the homogeneous class.
Usage

awcoord(xd, clvecd, clnum=1, mahal="square", method="classical",
        clweight=switch(method, classical=FALSE, TRUE),
        alpha=0.99, subsample=0, countmode=1000, ...)

Arguments

xd      the data matrix; a numerical object which can be coerced to a matrix.
clvecd  integer vector of class numbers; length must equal nrow(xd).
clnum   integer. Number of the homogeneous class.
mahal   "md" or "square". If "md", the points are weighted by the square root of
        the alpha-quantile of the corresponding chi squared distribution over the roots
        of their Mahalanobis distance to the homogeneous class, unless this is smaller
        than 1. If "square" (which is recommended), the (originally squared) Mahalanobis
        distance and the unrooted quantile is used.
method  one of "mve", "mcd" or "classical". Covariance matrix used within the homoge-
        neous class and for the computation of the Mahalanobis distances. "mcd" and
        "mve" are robust covariance matrices as implemented in cov.rob. "classical"
        refers to the classical covariance matrix.
clweight logical. If FALSE, only the points of the heterogeneous class are weighted.
        This, together with method="classical", computes AWC as defined in Hennig
        (2003). If TRUE, all points are weighted. This, together with method="mcd",
        computes ARC as defined in Hennig (2003).
alpha   numeric between 0 and 1. The corresponding quantile of the chi squared dis-
        tribution is used for the downweighting of points. Points with a smaller Maha-
        lanobis distance to the homogeneous class get full weight.
subsample integer. If 0, all points are used. Else, only a subsample of subsample of
        the points is used.
countmode optional positive integer. Every countmode algorithm runs awcoord shows a
        message.
...      no effect

Details

The square root of the homogeneous classes covariance matrix is inverted by use of tdecomp, which
can be expected to give reasonable results for singular within-class covariance matrices.

Value

List with the following components

ev   eigenvalues in descending order.
units columns are coordinates of projection basis vectors. New points x can be pro-
      jected onto the projection basis vectors by x %% units
proj  projections of xd onto units.
Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References

See Also
plotcluster for straight forward discriminant plots. discrproj for alternatives. rFace for generation of the example data used below.

Examples
set.seed(4634)
face <- rFace(600,dMoNo=2,dNoEy=0)
grface <- as.integer(attr(face,"grouping"))
awcf <- awcoord(face,grface==1)
# awcf2 <- ancoord(face,grface==1, method="mcd")
plot(awcf$proj,col=1+(grface==1))
# plot(awcf2$proj,col=1+(grface==1))
# ...done in one step by function plotcluster.

Description
Computes Bhattacharyya discriminant projection coordinates as described in Fukunaga (1990), p. 455 ff.

Usage
batcoord(xd, clvecd, clnum=1, dom="mean")
batvarcoord(xd, clvecd, clnum=1)

Arguments
xd the data matrix; a numerical object which can be coerced to a matrix.
clvecd integer or logical vector of class numbers; length must equal nrow(xd).
clnum integer, one of the values of clvecd, if this is an integer vector. Bhattacharyya projections can only be computed if there are only two classes in the dataset. clnum is the number of one of the two classes. All the points indicated by other values of clvecd are interpreted as the second class.
The discriminant coordinate for the group means is computed as the first projection direction by `discrcoord` (option `pool="equal"`; both classes have the same weight for computing the within-class covariance matrix). Then the data is projected into a subspace orthogonal (w.r.t. the within-class covariance) to the discriminant coordinate, and the projection coordinates to maximize the differences in variance are computed.

`dom="variance"` means that the projection coordinates maximizing the difference in variances are computed. Then they are ordered with respect to the Bhattacharyya distance, which takes also the mean differences into account. Both procedures are implemented as described in Fukunaga (1990).

**Details**

`batvarcoord` computes the optimal projection coordinates with respect to the difference in variances. `batcoord` combines the differences in mean and variance as explained for the argument `dom`.

**Value**

`batcoord` returns a list with the components `ev`, `rev`, `units`, `proj`. `batvarcoord` returns a list with the components `ev`, `rev`, `units`, `proj`, `w`, `S1`, `S2`.

- **ev**: vector of eigenvalues. If `dom="mean"`, then first eigenvalue from `discrcoord`. Further eigenvalues are of $S_i^{-1}S_2$, where $S_i$ is the covariance matrix of class $i$. For `batvarcoord` or if `dom="variance"`, all eigenvalues come from $S_i^{-1}S_2$ and are ordered by `rev`.


- **units**: columns are coordinates of projection basis vectors. New points $x$ can be projected onto the projection basis vectors by $x \mathbf{E}^* \mathbf{units}$.

- **proj**: projections of $xd$ onto `units`.

- **w**: matrix $S_1^{-1}S_2$.

- **S1**: covariance matrix of the first class.

- **S2**: covariance matrix of the second class.

**Author(s)**

Christian Hennig &lt;c.hennig@ucl.ac.uk&gt; [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

**References**

bhattacharyya.dist

See Also

- `plotcluster` for straight forward discriminant plots.
- `discrcoord` for discriminant coordinates.
- `rFace` for generation of the example data used below.

Examples

```r
set.seed(4634)
face <- rFace(600, dMoNo=2, dNoEy=8)
grface <- as.integer(attr(face, "grouping"))
bcf2 <- batcoord(face, grface==2)
plot(bcf2$proj, col=1+(grface==2))
bcfv2 <- batcoord(face, grface==2, dom="variance")
plot(bcfv2$proj, col=1+(grface==2))
bcfvv2 <- batvarcoord(face, grface==2)
plot(bcfvv2$proj, col=1+(grface==2))
```

**bhattacharyya.dist**  
*Bhattacharyya distance between Gaussian distributions*

### Description

Computes Bhattacharyya distance between two multivariate Gaussian distributions. See Fukunaga (1990).

### Usage

`bhattacharyya.dist(mu1, mu2, Sigma1, Sigma2)`

### Arguments

- **mu1**  
  mean vector of component 1.
- **mu2**  
  mean vector of component 2.
- **Sigma1**  
  covariance matrix of component 1.
- **Sigma2**  
  covariance matrix of component 2.

### Value

The Bhattacharyya distance between the two Gaussian distributions.

### Note

Thanks to David Pinto for improving this function.

### Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)
bhattacharyya.matrix

References


Examples

```r
round(bhattacharyya.dist(c(1,1),c(2,5),diag(2),diag(2)),digits=2)
```

bhattacharyya.matrix  Matrix of pairwise Bhattacharyya distances

Description

Computes Bhattachryya distances for pairs of components given the parameters of a Gaussian mixture.

Usage

```r
bhattacharyya.matrix(muarray,Sigmaarray,ipairs="all", misclassification.bound=TRUE)
```

Arguments

- `muarray`: matrix of component means (different components are in different columns).
- `Sigmaarray`: three dimensional array with component covariance matrices (the third dimension refers to components).
- `ipairs`: "all" or list of vectors of two integers. If `ipairs="all"`, computations are carried out for all pairs of components. Otherwise, `ipairs` gives the pairs of components for which computations are carried out.
- `misclassification.bound`: logical. If `TRUE`, upper bounds for misclassification probabilities \( \exp(-b) \) are given out instead of the original Bhattacharyya distances \( b \).

Value

A matrix with Bhattacharyya distances (or derived misclassification bounds, see above) between pairs of Gaussian distributions with the provided parameters. If `ipairs!="all"`, the Bhattacharyya distance and the misclassification bound are given as `NA` for pairs not included in `ipairs`.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)
References


See Also

*bhattacharyya.dist*

Examples

```r
muarray <- cbind(c(0,0),c(0,0.1),c(10,10))
sigmaarray <- array(c(diag(2),diag(2),diag(2)),dim=c(2,2,3))
bhattacharyya.matrix(muarray,sigmaarray,ipairs=list(c(1,2),c(2,3)))
```

calinhsara

**Calinski-Harabasz index**

Description

Calinski-Harabasz index for estimating the number of clusters, based on an observations/variables-matrix here. A distance based version is available through `cluster.stats`.

Usage

```r
calinhsara(x,clustering,cn=max(clustering))
```

Arguments

- `x`: data matrix or data frame.
- `cn`: integer. Number of clusters.

Value

Calinski-Harabasz statistic, which is \((n-cn)\times\text{sum(diag}(B))/((cn-1)\times\text{sum(diag}(W)))\). B being the between-cluster means, and W being the within-clusters covariance matrix.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche](http://www.homepages.ucl.ac.uk/~ucakche)

References

See Also

cluster.stats

Examples

```r
set.seed(98765)
iriss <- iris[sample(150,20),-5]
km <- kmeans(iriss,3)
round(calinhara(iriss,km$cluster),digits=2)
```

Description

Generates tuning constants `ca` for `fixreg` dependent on the number of points and variables of the dataset.

Only thought for use in `fixreg`.

Usage

`can(n, p)`

Arguments

- `n` positive integer. Number of points.
- `p` positive integer. Number of independent variables.

Details

The formula is \(3 + 33/(n \times 2^{-(p-1)/2})^{1/3} + 2900000/(n \times 2^{-(p-1)/2})^{3}\). For justification cf. Hennig (2002).

Value

A number.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

References

See Also

fixreg

Examples

```
can(429,3)
```

---

**Description**

Recodes a dataset with nominal variables so that the nominal variables are replaced by binary variables for the categories.

**Usage**

```
cat2bin(x, categorical=NULL)
```

**Arguments**

- `x` data matrix or data frame. The data need to be organised case-wise, i.e., if there are categorical variables only, and 15 cases with values c(1,1,2) on the 3 variables, the data matrix needs 15 rows with values 1 1 2. (Categorical variables could take numbers or strings or anything that can be coerced to factor levels as values.)

- `categorical` vector of numbers of variables to be recoded.

**Value**

A list with components

- `data` data matrix with variables specified in `categorical` replaced by 0-1 variables, one for each category.

- `variableinfo` list of lists. One list for every variable in the original dataset, with four components each, namely type ("categorical" or "not recoded"), levels (levels of nominal recoded variables in order of binary variable in output dataset), `ncat` (number of categories for recoded variables), `varnum` (number of variables in output dataset belonging to this original variable).

**Author(s)**

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche](http://www.homepages.ucl.ac.uk/~ucakche)

**See Also**

`discrete.recode`
Examples

```r
set.seed(77655)
v1 <- rnorm(20)
v2 <- rnorm(20)
d1 <- sample(1:5, 20, replace=TRUE)
d2 <- sample(1:4, 20, replace=TRUE)
data <- cbind(v1, v2, d1, d2)
lc <- cat2bin(data, categorical=3:4)
```

**Description**

CDbw-index for cluster validation, as defined in Halkidi and Vazirgiannis (2008), Halkidi et al. (2015).

**Usage**

```r
cdbw(x, clustering, r=10, s=seq(0.1, 0.8, by=0.1),
     clusterstdev=TRUE, trace=FALSE)
```

**Arguments**

- `x`: something that can be coerced into a numerical matrix. Euclidean dataset.
- `clustering`: vector of integers with length = `nrow(x)`; indicating the cluster for each observation.
- `r`: integer. Number of cluster border representatives.
- `s`: numerical vector of shrinking factors (between 0 and 1).
- `clusterstdev`: logical. If `TRUE`, the neighborhood radius for intra-cluster density is the within-cluster estimated squared distance from the mean of the cluster; otherwise it is the average of these over all clusters.
- `trace`: logical. If `TRUE`, results are printed for the steps to compute the index.

**Value**

List with components (see Halkidi and Vazirgiannis (2008), Halkidi et al. (2015) for details)

- `cdbw`: value of CDbw index (the higher the better).
- `cohesion`: cohesion.
- `compactness`: compactness.
- `sep`: separation.
Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


Examples

```r
options(digits=3)
iriss <- as.matrix(iris[c(3:51:55,101:105),,-5])
irisc <- as.numeric(iris[c(3:51:55,101:105),5])
cdbw(iriss,irisc)
```

---

**classifdist**  
*Classification of unclustered points*

Description

Various methods for classification of unclustered points from clustered points for use within functions nselectboot and prediction.strength.

Usage

```r
classifdist(cdist,clustering,
               method="averagedist",
               centroids=NULL,nnk=1)

classifnp(data,clustering,
               method="centroid",cdist=NULL,
               centroids=NULL,nnk=1)
```

Arguments

- `cdist` dissimilarity matrix or dist-object. Necessary for classifdist but optional for classifnp and there only used if method="averagedist" (if not provided, dist is applied to data).
- `data` something that can be coerced into a an n*p-data matrix.
- `clustering` integer vector. Gives the cluster number (between 1 and k for k clusters) for clustered points and should be -1 for points to be classified.
- `method` one of "averagedist", "centroid", "qda", "knn". See details.
classifdist

for classifnp a k times p matrix of cluster centroids. For classifdist a vector of numbers of centroid objects as provided by pam. Only used if method="centroid"; in that case mandatory for classifdist but optional for classifnp, where cluster mean vectors are computed if centroids=NULL.

nnk number of nearest neighbours if method="knn".

Details

classifdist is for data given as dissimilarity matrix, classifnp is for data given as n times p data matrix. The following methods are supported:

"centroid" assigns observations to the cluster with closest cluster centroid as specified in argument centroids (this is associated to k-means and pam/clara-clustering).

"averagedist" assigns to the cluster to which an observation has the minimum average dissimilarity to all points in the cluster (this is associated to average linkage clustering).

"qda" only in classifnp. Classifies by quadratic discriminant analysis (this is associated to Gaussian clusters with flexible covariance matrices), calling qda with default settings. If qda gives an error (usually because a class was too small), lda is used.

"knn" classifies by nnk nearest neighbours (for nnk=1, this is associated with single linkage clustering). Calls knn in classifnp.

Value

An integer vector giving cluster numbers for all observations; those for the observations already clustered in the input are the same as in the input.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

See Also

prediction.strength, nselectboot

Examples

s.set.seed(20000)
x1 <- rnorm(50)
y <- rnorm(100)
x2 <- rnorm(40,mean=20)
x3 <- rnorm(10,mean=25,sd=100)
x <- cbind(c(x1,x2,x3),y)
truec <- c(rep(1,50),rep(2,40),rep(3,10))
topredict <- c(1,2,51,52,91)
clumin <- truec
clumin[topredict] <- -1

classifnp(x,clumin, method="averagedist")
classifnp(x,clumin, method="qda")
**clucols**

*Sets of colours and symbols for cluster plotting*

**Description**

clucols gives out a vector of different random colours. clugrey gives out a vector of equidistant grey scales. clusym is a vector of different symbols starting from "1", "2", ...

**Usage**

```r
clucols(i, seed=NULL)  
clugrey(i, max=0.9)  
clusym
```

**Arguments**

- `i` integer. Length of output vector (number of clusters).
- `seed` integer. Random seed.
- `max` between 0 and 1. Maximum grey scale value, see `grey` (close to 1 is bright).

**Value**

clucols gives out a vector of different random colours. clugrey gives out a vector of equidistant grey scales. clusym is a vector of different characters starting from "1", "2", ...

**Author(s)**

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche](http://www.homepages.ucl.ac.uk/~ucakche)

**Examples**

```r
set.seed(112233)  
require(MASS)  
require(flexmix)  
data(Cars93)  
Cars934 <- Cars93[,c(3,5,8,10)]  
cc <- discrete.recode(Cars934,xvarsorted=FALSE,continuous=c(2,3),discrete=c(1,4))  
fcc <- flexmix(cc$data=1,k=3, model=lcmixed(continuous=2,discrete=2,pdim=c(6,3),diagonal=TRUE))  
plot(Cars934[,c(2,3)],col=clucols(3)[fcc$cluster],pch=clusym[fcc$cluster])
```
**clujaccard**

*Jaccard similarity between logical vectors*

**Description**

Jaccard similarity between logical or 0-1 vectors: \( \text{sum}(c1 \& c2)/\text{sum}(c1 | c2) \).

**Usage**

\[
\text{clujaccard}(c1, c2, \text{zerobyzero}=\text{NA})
\]

**Arguments**

- **c1**: logical or 0-1-vector.
- **c2**: logical or 0-1-vector (same length).
- **zerobyzero**: result if \( \text{sum}(c1 | c2)=0 \).

**Value**

Numeric between 0 and 1.

**Author(s)**

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

**Examples**

```r
  c1 <- rep(TRUE, 10)
  c2 <- c(FALSE, rep(TRUE, 9))
  clujaccard(c1, c2)
```

**clusexpect**

*Expected value of the number of times a fixed point cluster is found*

**Description**

A rough approximation of the expectation of the number of times a well separated fixed point cluster (FPC) of size \( n \) is found in \( ir \) fixed point iterations of \texttt{fixreg}.

**Usage**

\[
\text{clusexpect}(n, p, cn, ir)
\]
Arguments

- **n** positive integer. Total number of points.
- **p** positive integer. Number of independent variables.
- **cn** positive integer smaller or equal to \( n \). Size of the FPC.
- **ir** positive integer. Number of fixed point iterations.

Details

The approximation is based on the assumption that a well separated FPC is found iff all \( p+2 \) points of the initial configuration come from the FPC. The value is \( ir \) times the probability for this. For a discussion of this assumption cf. Hennig (2002).

Value

A number.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

References


See Also

fixreg

Examples

```r
round(clusexpect(500,4,150,2000),digits=2)
```

### cluster.stats

*Cluster validation statistics*

Description

Computes a number of distance based statistics, which can be used for cluster validation, comparison between clusterings and decision about the number of clusters: cluster sizes, cluster diameters, average distances within and between clusters, cluster separation, biggest within cluster gap, average silhouette widths, the Calinski and Harabasz index, a Pearson version of Hubert’s gamma coefficient, the Dunn index and two indexes to assess the similarity of two clusterings, namely the corrected Rand index and Meila’s VI.
Usage

c_cluster.stats(d = NULL, clustering, alt.clustering = NULL,
      noisecluster=FALSE,
      silhouette = TRUE, G2 = FALSE, G3 = FALSE,
      wgap=TRUE, sepindex=TRUE, sepprob=0.1,
      sepwithinnoise=TRUE,
      compareonly = FALSE,
      aggregateonly = FALSE)

Arguments

d  a distance object (as generated by dist) or a distance matrix between cases.
clustering  an integer vector of length of the number of cases, which indicates a clustering.
The clusters have to be numbered from 1 to the number of clusters.
alt.clustering  an integer vector such as for clustering, indicating an alternative clustering. If provided, the corrected Rand index and Meila's VI for clustering vs. alt.clustering are computed.
noisecluster  logical. If TRUE, it is assumed that the largest cluster number in clustering denotes a 'noise class', i.e. points that do not belong to any cluster. These points are not taken into account for the computation of all functions of within and between cluster distances including the validation indexes.
silhouette  logical. If TRUE, the silhouette statistics are computed, which requires package cluster.
G2  logical. If TRUE, Goodman and Kruskal's index G2 (cf. Gordon (1999), p. 62) is computed. This executes lots of sorting algorithms and can be very slow (it has been improved by R. Francois - thanks!)
G3  logical. If TRUE, the index G3 (cf. Gordon (1999), p. 62) is computed. This executes sort on all distances and can be extremely slow.
wgap  logical. If TRUE, the widest within-cluster gaps (largest link in within-cluster minimum spanning tree) are computed. This is used for finding a good number of clusters in Hennig (2013).
sepindex  logical. If TRUE, a separation index is computed, defined based on the distances for every point to the closest point not in the same cluster. The separation index is then the mean of the smallest proportion sepprob of these. This allows to formalise separation less sensitive to a single or a few ambiguous points. The output component corresponding to this is sindex, not separation! This is used for finding a good number of clusters in Hennig (2013).
sepprob  numerical between 0 and 1, see sepindex.
sepwithinnoise  logical. If TRUE and sepindex and noisecluster are both TRUE, the noise points are incorporated as cluster in the separation index (sepindex) computation. Also they are taken into account for the computation for the minimum cluster separation.
compareonly  logical. If TRUE, only the corrected Rand index and Meila’s VI are computed and given out (this requires alt.clustering to be specified).
aggregateonly  logical. If TRUE (and not compareonly), no clusterwise but only aggregated information is given out (this cuts the size of the output down a bit).
cluster.stats returns a list containing the components `n`, `cluster.number`, `cluster.size`, `min.cluster.size`, `noisen`, `diameter`, `average.distance`, `median.distance`, `separation`, `average.toother`, `separation.matrix`, `ave.between.matrix`, `average.between`, `average.within`, `n.between`, `n.within`, `max.diameter`, `min.separation`, `within.cluster.ss`, `clus.avg.silwidths`, `avg.silwidth`, `g2`, and `g3` except if `compareonly=TRUE`, in which case only the last two components are computed.

- `n`: number of cases.
- `cluster.number`: number of clusters.
- `cluster.size`: vector of cluster sizes (number of points).
- `min.cluster.size`: size of smallest cluster.
- `noisen`: number of noise points, see argument `noisecluster` (noisen=0 if noisecluster=FALSE).
- `diameter`: vector of cluster diameters (maximum within cluster distances).
- `average.distance`: vector of clusterwise within cluster average distances.
- `median.distance`: vector of clusterwise within cluster distance medians.
- `separation`: vector of clusterwise minimum distances of a point in the cluster to a point of another cluster.
- `average.toother`: vector of clusterwise average distances of a point in the cluster to the points of other clusters.
- `separation.matrix`: matrix of separation values between all pairs of clusters.
- `ave.between.matrix`: matrix of mean dissimilarities between points of every pair of clusters.
- `average.between`: average distance between clusters.
- `average.within`: average distance within clusters.
- `n.between`: number of distances between clusters.
- `n.within`: number of distances within clusters.
- `max.diameter`: maximum cluster diameter.
- `min.separation`: minimum cluster separation.
- `within.cluster.ss`: a generalisation of the within clusters sum of squares (k-means objective function), which is obtained if `d` is a Euclidean distance matrix. For general distance measures, this is half the sum of the within cluster squared dissimilarities divided by the cluster size.
- `clus.avg.silwidths`: vector of cluster average silhouette widths. See `silhouette`.
- `avg.silwidth`: average silhouette width. See `silhouette`.
cluster.stats

pearsongamma  correlation between distances and a 0-1-vector where 0 means same cluster, 1 means different clusters. "Normalized gamma" in Halkidi et al. (2001).
dunn  minimum separation / maximum diameter. Dunn index, see Halkidi et al. (2002).
dunn2  minimum average dissimilarity between two cluster / maximum average within cluster dissimilarity, another version of the family of Dunn indexes.
entropy  entropy of the distribution of cluster memberships, see Meila(2007).
wbnratio  average.within/average.between.
ch  Calinski and Harabasz index (Calinski and Harabasz 1974, optimal in Milligan and Cooper 1985; generalised for dissimilarities in Hennig and Liao 2013).
cwdegap  vector of widest within-cluster gaps.
widestgap  widest within-cluster gap.
sindex  separation index, see argument sepindex.
corrected.rand  corrected Rand index (if alt.clustering has been specified), see Gordon (1999, p. 198).
vi  variation of information (VI) index (if alt.clustering has been specified), see Meila (2007).

Note
Because cluster.stats processes a full dissimilarity matrix, it isn’t suitable for large data sets. You may consider distcritmulti in that case.

Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References

cluster.varstats

Variablewise statistics for clusters

Description

This function gives some helpful variable-wise information for cluster interpretation, given a clustering and a data set. The output object contains some tables. For categorical variables, tables compare clusterwise distributions with overall distributions. Continuous variables are categorised for this.

If desired, tables, histograms, some standard statistics of continuous variables and validation plots as available through discrproj (Hennig 2004) are given out on the fly.

Usage

cluster.varstats(clustering, vardata, contdata=vardata,
clusterwise=TRUE,
tablevar=NULL, catvar=NULL,
quantvar=NULL, catvarcats=10,
proportions=FALSE,
projmethod="none", msize=ncol(contdata)+2,
ask=TRUE, rangefactor=1)

## S3 method for class 'varwisetables'
print(x, digits=3,...)

Arguments

clustering vector of integers. Clustering (needs to be in standard coding, 1,2,...).
vardata data matrix or data frame of which variables are summarised.
cluster.varstats

contdata variable matrix or data frame, normally all or some variables from vardata, on which cluster visualisation by projection methods is performed unless projmethod="none". It should make sense to interpret these variables in a quantitative (interval-scaled) way.

clusterwise logical. If FALSE, only the output tables are computed but no more detail and graphs are given on the fly.

tablevar vector of integers. Numbers of variables treated as categorical (i.e., no histograms and statistics, just tables) if clusterwise=TRUE. Note that an error will be produced by factor type variables unless they are declared as categorical here.

catvar vector of integers. Numbers of variables to be categorised by proportional quantiles for table computation. Recommended for all continuous variables.

quantvar vector of integers. Variables for which means, standard deviations and quantiles should be given out if clusterwise=TRUE.

catvarcats integer. Number of categories used for categorisation of variables specified in quantvar.

proportions logical. If TRUE, output tables contain proportions, otherwise numbers of observations.

projmethod one of "none", "dc", "bc", "vbc", "mvdc", "adc", "awc" (recommended if not "none"), "arc", "nc", "wnc", "anc". Cluster validation projection method introduced in Hennig (2004), passed on as method argument in discrproj.

minsize integer. Projection is not carried out for clusters with fewer points than this. (If this is chosen smaller, it may lead to errors with some projection methods.)

ask logical. If TRUE, par(ask=TRUE) is set in the beginning to prompt the user before plots and par(ask=FALSE) in the end.

rangefactor numeric. Factor by which to multiply the range for projection plot ranges.

x an object of class "varwisetables", output object of cluster.varstats.

digits integer. Number of digits after the decimal point to print out.

... not used.

Value

An object of class "varwisetables", which is a list with a table for each variable, giving (categorised) marginal distributions by cluster.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche

References

Examples

```r
set.seed(112233)
options(digits=3)
require(MASS)
require(flexmix)
data(Cars93)
Cars934 <- Cars93[,c(3,5,8,10)]
cc <-
discrete.recode(Cars934,xvarsorted=FALSE,continuous=c(2,3),discrete=c(1,4))
fcc <- flexmix(cc$data=1,k=2,
model=lcmixed(continuous=2,discrete=2,pdim=c(6,3),diagonal=TRUE))
cv <-
cluster.varstats(fcc@cluster,Cars934, contdata=Cars934[,c(2,3)],
tablevar=c(1,4),catvar=c(2,3),quantvar=c(2,3),projmethod="awc",
ask=FALSE)
print(cv)
```

Description

Assessment of the clusterwise stability of a clustering of data, which can be cases*variables or dissimilarity data. The data is resampled using several schemes (bootstrap, subsetting, jittering, replacement of points by noise) and the Jaccard similarities of the original clusters to the most similar clusters in the resampled data are computed. The mean over these similarities is used as an index of the stability of a cluster (other statistics can be computed as well). The methods are described in Hennig (2007).

clusterboot is an integrated function that computes the clustering as well, using interface functions for various clustering methods implemented in R (several interface functions are provided, but you can implement further ones for your favourite clustering method). See the documentation of the input parameter clustermethod below.

Quite general clustering methods are possible, i.e. methods estimating or fixing the number of clusters, methods producing overlapping clusters or not assigning all cases to clusters (but declaring them as "noise"). Fuzzy clusterings cannot be processed and have to be transformed to crisp clusterings by the interface function.

Usage

```r
clusterboot(data,B=100, distances=(class(data)=="dist"),
bootmethod="boot",
bscompare=TRUE,
multipleboot=FALSE,
jittertuning=0.05, noisetuning=c(0.05,4),
subtuning=floor(nrow(data)/2),
clustermethod, noisemethod=FALSE, count=TRUE,
showplots=FALSE,dissolution=0.5,)
```
recover=0.75,seed=NULL,datatomatrix=TRUE,...)

## S3 method for class 'clboot'
print(x,statistics=c("mean","dissolution","recovery"),...)

## S3 method for class 'clboot'
plot(x,xlim=c(0,1),breaks=seq(0,1,by=0.05),...)

Arguments

data

by default something that can be coerced into a (numerical) matrix (data frames with non-numerical data are allowed when using datatomatrix=FALSE, see below). The data matrix - either an n*p-data matrix (or data frame) or an n*n-dissimilarity matrix (or dist-object).

B

integer. Number of resampling runs for each scheme, see bootmethod.

distances

logical. If TRUE, the data is interpreted as dissimilarity matrix. If data is a dist-object, distances=TRUE automatically, otherwise distances=FALSE by default. This means that you have to set it to TRUE manually if data is a dissimilarity matrix.

bootmethod

vector of strings, defining the methods used for resampling. Possible methods:

"boot": nonparametric bootstrap (precise behaviour is controlled by parameters bscompare and multipleboot).

"subset": selecting random subsets from the dataset. Size determined by subtuning.

"noise": replacing a certain percentage of the points by random noise, see noisetuning (note that this will not work if within-cluster.

"jitter" add random noise to all points, see jittertuning. (This didn’t perform well in Hennig (2007), but you may want to get your own experience.)

"bojit" nonparametric bootstrap first, and then adding noise to the points, see jittertuning.

Important: only the methods "boot" and "subset" work with dissimilarity data, or if datatomatrix=FALSE!

The results in Hennig (2007) indicate that "boot" is generally informative and often quite similar to "subset" and "bojit", while "noise" sometimes provides different information. Therefore the default (for distances=FALSE) is to use "boot" and "noise". However, some clustering methods may have problems with multiple points, which can be solved by using "bojit" or "subset" instead of "boot" or by multipleboot=FALSE below.

bscompare

logical. If TRUE, multiple points in the bootstrap sample are taken into account to compute the Jaccard similarity to the original clusters (which are represented by their "bootstrap versions", i.e., the points of the original cluster which also occur in the bootstrap sample). If a point was drawn more than once, it is in the "bootstrap version" of the original cluster more than once, too, if bscompare=TRUE. Otherwise multiple points are ignored for the computation of the Jaccard similarities. If multipleboot=FALSE, it doesn’t make a difference.

multipleboot

logical. If FALSE, all points drawn more than once in the bootstrap draw are only used once in the bootstrap samples.
jittertuning positive numeric. Tuning for the "jitter"-method. The noise distribution for jittering is a normal distribution with zero mean. The covariance matrix has the same Eigenvectors as that of the original data set, but the standard deviation along the principal directions is determined by the jittertuning-quantile of the distances between neighboring points projected along these directions.

noisetuning A vector of two positive numerics. Tuning for the "noise"-method. The first component determines the probability that a point is replaced by noise. Noise is generated by a uniform distribution on a hyperrectangle along the principal directions of the original data set, ranging from -noisetuning[2] to noisetuning[2] times the standard deviation of the data set along the respective direction. Note that only points not replaced by noise are considered for the computation of Jaccard similarities.

subtuning integer. Size of subsets for "subset".

clustermethod an interface function (the function name, not a string containing the name, has to be provided!). This defines the clustering method. See the "Details"-section for a list of available interface functions and guidelines how to write your own ones.

noisemethod logical. If TRUE, the last cluster is regarded as "noise cluster", which means that for computing the Jaccard similarity, it is not treated as a cluster. The noise cluster of the original clustering is only compared with the noise cluster of the clustering of the resampled data. This means that in the clusterboot-output (and plot), if points were assigned to the noise cluster, the last cluster number refers to it, and its Jaccard similarity values refer to comparisons with estimated noise components in resampled datasets only. (Some cluster methods such as trimmed k-means and mclustBIC produce such noise components.)

count logical. If TRUE, the resampling runs are counted on the screen.

showplots logical. If TRUE, a plot of the first two dimensions of the resampled data set (or the classical MDS solution for dissimilarity data) is shown for every resampling run. The last plot shows the original data set. Ignored if datatomatrix=FALSE.

dissolution numeric between 0 and 1. If the Jaccard similarity between the resampling version of the original cluster and the most similar cluster on the resampled data is smaller or equal to this value, the cluster is considered as "dissolved". Numbers of dissolved clusters are recorded.

recover numeric between 0 and 1. If the Jaccard similarity between the resampling version of the original cluster and the most similar cluster on the resampled data is larger than this value, the cluster is considered as "successfully recovered". Numbers of recovered clusters are recorded.

seed integer. Seed for random generator (fed into set.seed) to make results reproducible. If NULL, results depend on chance.

datatomatrix logical. If TRUE, data is coerced into a (numerical) matrix at the start of clusterboot. FALSE may be chosen for mixed type data including e.g. categorical factors (assuming that the chosen clustermethod allows for this). This disables some features of clusterboot, see parameters bootmethod and showplots.

... additional parameters for the clustermethods called by clusterboot. No effect in print.clboot and plot.clboot.
**clusterboot**

- **x**: object of class `clboot`.
- **statistics**: specifies in `print.clboot`, which of the three clusterwise Jaccard similarity statistics: "mean", "dissolution" (number of times the cluster has been dissolved) and "recovery" (number of times a cluster has been successfully recovered) is printed.
- **xlim**: transferred to `hist`.
- **breaks**: transferred to `hist`.

**Details**

Here are some guidelines for interpretation. There is some theoretical justification to consider a Jaccard similarity value smaller or equal to 0.5 as an indication of a "dissolved cluster", see Hennig (2008). Generally, a valid, stable cluster should yield a mean Jaccard similarity value of 0.75 or more. Between 0.6 and 0.75, clusters may be considered as indicating patterns in the data, but which points exactly should belong to these clusters is highly doubtful. Below average Jaccard values of 0.6, clusters should not be trusted. "Highly stable" clusters should yield average Jaccard similarities of 0.85 and above. All of this refers to bootstrap; for the other resampling schemes it depends on the tuning constants, though their default values should grant similar interpretations in most cases.

While `B=100` is recommended, smaller run numbers could give quite informative results as well, if computation times become too high.

Note that the stability of a cluster is assessed, but stability is not the only important validity criterion - clusters obtained by very inflexible clustering methods may be stable but not valid, as discussed in Hennig (2007). See `plotcluster` for graphical cluster validation.

Information about interface functions for clustering methods:

The following interface functions are currently implemented (in the present package; note that almost all of these functions require the specification of some control parameters, so if you use one of them, look up their common help page `kmeansCBI`) first:

- **kmeansCBI**: an interface to the function `kmeans` for k-means clustering. This assumes a cases*variables matrix as input.
- **hclustCBI**: an interface to the function `hclust` for agglomerative hierarchical clustering with optional noise cluster. This function produces a partition and assumes a cases*variables matrix as input.
- **hclusttreeCBI**: an interface to the function `hclust` for agglomerative hierarchical clustering. This function produces a tree (not only a partition; therefore the number of clusters can be huge!) and assumes a cases*variables matrix as input.
- **disthclustCBI**: an interface to the function `hclust` for agglomerative hierarchical clustering with optional noise cluster. This function produces a partition and assumes a dissimilarity matrix as input.
- **noisemclustCBI**: an interface to the function `mclustBIC` for normal mixture model based clustering. This assumes a cases*variables matrix as input. Warning: `mclustBIC` sometimes has problems with multiple points. It is recommended to use this only together with `multipleboot=FALSE`.
- **distnoisemclustCBI**: an interface to the function `mclustBIC` for normal mixture model based clustering. This assumes a dissimilarity matrix as input and generates a data matrix by multidimensional scaling first. Warning: `mclustBIC` sometimes has problems with multiple points. It is recommended to use this only together with `multipleboot=FALSE`.
clusterboot

claraCBI an interface to the functions pam and clara for partitioning around medoids. This can be used with cases*variables as well as dissimilarity matrices as input.

pamkCBI an interface to the function pamk for partitioning around medoids. The number of cluster is estimated by the average silhouette width. This can be used with cases*variables as well as dissimilarity matrices as input.

trimkmeansCBI an interface to the function trimkmeans for trimmed k-means clustering. This assumes a cases*variables matrix as input.

tclustCBI an interface to the function tclust in the tclust library for trimmed Gaussian clustering. This assumes a cases*variables matrix as input. Note that this function is not currently provided because the tclust package is only available in the CRAN archives, but the code is in the Examples-section of the kmeansCBI-help page.

disttrimkmeansCBI an interface to the function trimkmeans for trimmed k-means clustering. This assumes a dissimilarity matrix as input and generates a data matrix by multidimensional scaling first.

dbscanCBI an interface to the function dbscan for density based clustering. This can be used with cases*variables as well as dissimilarity matrices as input.

mahalCBI an interface to the function fixmahal for fixed point clustering. This assumes a cases*variables matrix as input.

mergenormCBI an interface to the function mergenormals for clustering by merging Gaussian mixture components.

speccCBI an interface to the function specc for spectral clustering.

You can write your own interface function. The first argument of an interface function should preferably be a data matrix (of class "matrix", but it may be a symmetrical dissimilarity matrix). It can be a data frame, but this restricts some of the functionality of clusterboot, see above. Further arguments can be tuning constants for the clustering method. The output of an interface function should be a list containing (at least) the following components:

result clustering result, usually a list with the full output of the clustering method (the precise format doesn’t matter); whatever you want to use later.

nc number of clusters. If some points don’t belong to any cluster but are declared as “noise”, nc includes the noise cluster, and there should be another component ncc1, being the number of clusters not including the noise cluster (note that it is not mandatory to define a noise component if not all points are assigned to clusters, but if you do it, the stability of the noise cluster is assessed as well.)

clusterlist this is a list consisting of a logical vectors of length of the number of data points (n) for each cluster, indicating whether a point is a member of this cluster (TRUE) or not. If a noise cluster is included, it should always be the last vector in this list.

partition an integer vector of length n, partitioning the data. If the method produces a partition, it should be the clustering. This component is only used for plots, so you could do something like rep(1, n) for non-partitioning methods. If a noise cluster is included, nc=ncc1+1 and the noise cluster is cluster no. nc.

clustermethod a string indicating the clustering method.
clusterboot

Value

clusterboot returns an object of class "clboot", which is a list with components result, partition, nc, clustermethod, b, noisemethod, bootmethod, multipleboot, dissolution, recover, subsetresult, subsetmean, etc.

result clustering result; full output of the selected clustermethod for the original data set.

partition partition parameter of the selected clustermethod (note that this is only meaningful for partitioning clustering methods).

nc number of clusters in original data (including noise component if noisemethod=TRUE).

nccl number of clusters in original data (not including noise component if noisemethod=TRUE).

clustermethod, B, noisemethod, bootmethod, multipleboot, dissolution, recover input parameters, see above.

bootresult matrix of Jaccard similarities for bootmethod="boot". Rows correspond to clusters in the original data set. Columns correspond to bootstrap runs.

bootmean clusterwise means of the bootresult.

bootbrd clusterwise number of times a cluster has been dissolved.

bootrecover clusterwise number of times a cluster has been successfully recovered.

subsetresult, subsetmean, etc. same as bootresult, bootmean, etc., but for the other resampling methods.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


See Also
dist, interface functions: kmeansCBI, hclustCBI, hclusttreeCBI, disthclustCBI, noisemclustCBI, distnoisemclustCBI, claraCBI, pamkCBI, trimkmeansCBI, disttrimkmeansCBI, dbscanCBI, mahalCBI

Examples

options(digits=3)
set.seed(20000)
face <- rFace(50,dMoNo=2,dNoEy=0,p=2)
cfl <- clusterboot(face,B=3,bootmethod=
  c("boot","noise","jitter"),clustermethod=kmeansCBI,
  krange=5,seed=15555)

print(cfl)
plot(cf1)

cf2 <- clusterboot(dist(face), B=3, bootmethod=  
  "subset", clustermethod=disthclustCBI,  
  k=5, cut="number", method="average", showplots=TRUE, seed=15555)
print(cf2)
d1 <- c("a", "b", "a", "c")
d2 <- c("a", "a", "a", "b")
dx <- as.data.frame(cbind(d1, d2))
cpx <- clusterboot(dx, k=2, B=10, clustermethod=claraCBI,  
  multipleboot=TRUE, usepam=TRUE, datatomatrix=FALSE)
print(cpx)

---

**cmahal**  
*Generation of tuning constant for Mahalanobis fixed point clusters.*

**Description**

Generates tuning constants ca for **fixmahal** dependent on the number of points and variables of the current fixed point cluster (FPC).

This is experimental and only thought for use in **fixmahal**.

**Usage**

`cmahal(n, p, nmin, cmin, nc1, c1 = cmin, q = 1)`

**Arguments**

- `n`  
  positive integer. Number of points.
- `p`  
  positive integer. Number of variables.
- `nmin`  
  integer larger than 1. Smallest number of points for which ca is computed. For smaller FPC sizes, ca is set to the value for nmin.
- `cmin`  
  positive number. Minimum value for ca.
- `nc1`  
  positive integer. Number of points at which ca=c1.
- `c1`  
  positive numeric. Tuning constant for cmahal. Value for ca for FPC size equal to nc1.
- `q`  
  numeric between 0 and 1. 1 for steepest possible descent of ca as function of the FPC size. Should presumably always be 1.

**Details**

Some experiments suggest that the tuning constant ca should decrease with increasing FPC size and increase with increasing p in **fixmahal**. This is to prevent too small meaningless FPCs while maintaining the significant larger ones. cmahal with q=1 computes ca in such a way that as long as ca>cmin, the decrease in n is as steep as possible in order to maintain the validity of the convergence theorem in Hennig and Christlieb (2002).
**Value**
A numeric vector of length \( n \), giving the values for \( ca \) for all FPC sizes smaller or equal to \( n \).

**Author(s)**
Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

**References**

**See Also**
fixmahal

**Examples**
```r
plot(1:100, cmahal(100, 3, nmin=5, cmin=qchisq(0.99, 3), nc1=90),
     xlab="FPC size", ylab="cmahal")
```

---

**con.comp**

*Connectivity components of an undirected graph*

**Description**
Computes the connectivity components of an undirected graph from a matrix giving the edges.

**Usage**
con.comp(comat)

**Arguments**

- **comat**: a symmetric logical or 0-1 matrix, where \( \text{comat}[i,j]=\text{TRUE} \) means that there is an edge between vertices \( i \) and \( j \). The diagonal is ignored.

**Details**
The "depth-first search" algorithm of Cormen, Leiserson and Rivest (1990, p. 477) is used.

**Value**
An integer vector, giving the number of the connectivity component for each vertex.

**Author(s)**
Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)
References


See Also

`hclust`, `cutree` for cutted single linkage trees (often equivalent).

Examples

```r
set.seed(1000)
> x <- rnorm(20)
m <- matrix(0,nrow=20,ncol=20)
for(i in 1:20)
  for(j in 1:20)
    m[i,j] <- abs(x[i]-x[j])
d <- m<=0.2
cc <- con.comp(d)
max(cc) # number of connectivity components
plot(x,cc)
# The same should be produced by
# cutree(hclust(as.dist(m),method="single"),h=0.2).
```

---

**confusion**

*Misclassification probabilities in mixtures*

Description

Estimates a misclassification probability in a mixture distribution between two mixture components from estimated posterior probabilities regardless of component parameters, see Hennig (2010).

Usage

```r
confusion(z,pro,i,j,adjustprobs=FALSE)
```

Arguments

- `z` matrix of posterior probabilities for observations (rows) to belong to mixture components (columns), so entries need to sum up to 1 for each row.
- `pro` vector of component proportions, need to sum up to 1.
- `i` integer. Component number.
- `j` integer. Component number.
- `adjustprobs` logical. If TRUE, probabilities are initially standardised so that those for components i and j add up to one (i.e., if they were the only components).
Value

Estimated probability that an observation generated by component $j$ is classified to component $i$ by maximum a posteriori rule.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


Examples

```r
set.seed(12345)
m <- rpois(20,lambda=5)
dim(m) <- c(5,4)
pro <- apply(m,2,sum)
pro <- pro/sum(pro)
m <- m/apply(m,1,sum)
round(confusion(m,pro,1,2),digits=2)
```

Description

Returns a list containing estimates of the weighted covariance matrix and the mean of the data, and optionally of the (weighted) correlation matrix. The covariance matrix is divided by the sum of the weights, corresponding to $n$ and the ML-estimator in the case of equal weights, as opposed to $n-1$ for `cov.wt`.

Usage

```r
cov.wml(x, wt = rep(1/nrow(x), nrow(x)), cor = FALSE, center = TRUE)
```

Arguments

- **x**: a matrix or data frame. As usual, rows are observations and columns are variables.
- **wt**: a non-negative and non-zero vector of weights for each observation. Its length must equal the number of rows of $x$.
- **cor**: A logical indicating whether the estimated correlation weighted matrix will be returned as well.
- **center**: Either a logical or a numeric vector specifying the centers to be used when computing covariances. If TRUE, the (weighted) mean of each variable is used, if FALSE, zero is used. If center is numeric, its length must equal the number of columns of $x$. 

---

**cov.wml**

*Weighted Covariance Matrices (Maximum Likelihood)*

---
Value

A list containing the following named components:

- **cov**: the estimated (weighted) covariance matrix.
- **center**: an estimate for the center (mean) of the data.
- **n.obs**: the number of observations (rows) in \( x \).
- **wt**: the weights used in the estimation. Only returned if given as an argument.
- **cor**: the estimated correlation matrix. Only returned if ‘cor’ is ‘TRUE’.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

See Also

cov.wt, cov.var

Examples

```r
x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
y <- c(1, 2, 3, 8, 7, 6, 5, 8, 9, 10)
cov.wt(cbind(x, y), wt=c(0, 0, 0, 1, 1, 1, 1, 0, 0))
cov.wt(cbind(x, y), wt=c(0, 0, 0, 1, 1, 1, 1, 0, 0))
```

---

cvnn: *Cluster validation based on nearest neighbours*

Description

Cluster validity index based on nearest neighbours as defined in Liu et al. (2013) with a correction explained in Halkidi et al. (2015).

Usage

cvnn(d=NULL, clusterings, k=5)

Arguments

dissimilarity matrix or dist-object.

- **clusterings**: list of vectors of integers with length =nrow(d); indicating the cluster for each observation for several clusterings (list elements) to be compared.

- **k**: integer. Number of nearest neighbours.
Value

List with components (see Liu et al. (2013), Halkidi et al. (2015) for details)

- **cvnnindex**: vector of index values for the various clusterings, see Liu et al. (2013), the lower the better.
- **sep**: vector of separation values.
- **comp**: vector of compactness values.

Refernce(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

Examples

```r
options(digits=3)
iriss <- as.matrix(iris[c(1:10,51:55,101:105),-5])
irisc <- as.numeric(iris[c(1:10,51:55,101:105),5])
print(cvnn(dist(iriss),list(irisc,rep(1:4,5))))
```

---

### cweight

*Weight function for AWC*

**Description**

For use in awcoord only.

**Usage**

```r
cweight(x, ca)
```

**Arguments**

- **x**: numerical.
- **ca**: numerical.

**Value**

`ca/x` if smaller than 1, else 1.
Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

See Also
awcoord

Examples

cweight(4,1)

dbscan

DBSCAN density reachability and connectivity clustering

Description
Generates a density based clustering of arbitrary shape as introduced in Ester et al. (1996).

Usage
dbscan(data, eps, MinPts = 5, scale = FALSE, method = c("hybrid", "raw", "dist"), seeds = TRUE, showplot = FALSE, countmode = NULL)

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

## S3 method for class 'dbscan'
plot(x, data, ...)

## S3 method for class 'dbscan'
predict(object, data, newdata = NULL, predict.max=1000, ...)

Arguments
data data matrix, data.frame, dissimilarity matrix or dist-object. Specify method="dist" if the data should be interpreted as dissimilarity matrix or object. Otherwise Euclidean distances will be used.
eps Reachability distance, see Ester et al. (1996).
MinPts Reachability minimum no. of points, see Ester et al. (1996).
scale scale the data if TRUE.
method "dist" treats data as distance matrix (relatively fast but memory expensive), "raw" treats data as raw data and avoids calculating a distance matrix (saves memory but may be slow), "hybrid" expects also raw data, but calculates partial distance matrices (very fast with moderate memory requirements).
seeds FALSE to not include the isseed-vector in the dbscan-object.
showplot 0 = no plot, 1 = plot per iteration, 2 = plot per subiteration.
countmode NULL or vector of point numbers at which to report progress.
**Details**

Clusters require a minimum no of points (MinPts) within a maximum distance (eps) around one of its members (the seed). Any point within eps around any point which satisfies the seed condition is a cluster member (recursively). Some points may not belong to any clusters (noise).

We have clustered a 100,000 x 2 dataset in 40 minutes on a Pentium M 1600 MHz.

`printNdbscan` shows a statistic of the number of points belonging to the clusters that are seeds and border points.

`plotNdbscan` distinguishes between seed and border points by plot symbol.

**Value**

`predictNdbscan` gives out a vector of predicted clusters for the points in `newdata`.

`dbscan` gives out an object of class `dbscan` which is a LIST with components:

- `cluster`: integer vector coding cluster membership with noise observations (singletons) coded as 0
- `isseed`: logical vector indicating whether a point is a seed (not border, not noise)
- `eps`: parameter eps
- `MinPts`: parameter MinPts

**Note**

this is a simplified version of the original algorithm (no K-D-trees used), thus we have \( o(n^2) \) instead of \( o(n \times log(n)) \)

**Author(s)**

Jens Oehlschlaegel, based on a draft by Christian Hennig.

**References**

Examples

```r
set.seed(665544)
n <- 600
x <- cbind(runif(10, 0, 10)+rnorm(n, sd=0.2), runif(10, 0, 10)+rnorm(n, sd=0.2))
par(bg="grey40")
ds <- dbscan(x, 0.2)
# run with showplot=1 to see how dbscan works.
ds
plot(ds, x)
x2 <- matrix(0,nrow=4,ncol=2)
x2[1,] <- c(5,2)
x2[2,] <- c(8,3)
x2[3,] <- c(4,4)
x2[4,] <- c(9,9)
predict(ds, x, x2)
```

```r
n <- 600
x <- cbind((1:3)+rnorm(n, sd=0.2), (1:3)+rnorm(n, sd=0.2))
# Not run, but results from my machine are 0.105 - 0.068 - 0.255:
# system.time(ds <- dbscan(x, 0.3, countmode=NULL, method="raw")[3]
# system.time(dsb <- dbscan(x, 0.3, countmode=NULL, method="hybrid")[3]
# system.time(dsc <- dbscan(dist(x), 0.3, countmode=NULL, method="dist")[3]
```

dipp.tantrum

Simulates p-value for dip test

Description

Simulates p-value for dip test (see dip) in the way suggested by Tantrum, Murua and Stuetzle (2003) from the closest unimodal distribution determined by kernel density estimation with bandwidth chosen so that the density just becomes unimodal. This is less conservative (and in fact sometimes anti-conservative) than the values from dip.test.

Usage

dipp.tantrum(xdata, d, M=100)

Arguments

- **xdata** numeric vector. One-dimensional dataset.
- **d** numeric. Value of dip statistic.
- **M** integer. Number of artificial datasets generated in order to estimate the p-value.
**Value**

List with components

- `p.value`: approximated p-value.
- `bw`: borderline unimodality bandwith in `density` with default settings.
- `dv`: vector of dip statistic values from simulated artificial data.

**Author(s)**

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

**References**


**Examples**

```r
# not run, requires package diptest
# x <- runif(100)
# d <- dip(x)
# dt <- dipp.tantrum(x,d,M=10)
```

**Description**

Diptest (Hartigan and Hartigan, 1985, see `dip`) for data projected in discriminant coordinate separating optimally two class means (see `discrcoord`) as suggested by Tantrum, Murua and Stuetzle (2003).

**Usage**

```r
diptest.multi(xdata,class,pvalue="uniform",M=100)
```

**Arguments**

- `class`: vector of integers giving class numbers for observations.
- `pvalue`: "uniform" or "tantrum". Defines whether the p-value is computed from a uniform null model as suggested in Hartigan and Hartigan (1985, using `dip.test`) or as suggested in Tantrum et al. (2003, using dipp.tantrum).
- `M`: integer. Number of artificial datasets generated in order to estimate the p-value if `pvalue="tantrum"`. 
Value

The resulting p-value.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


Examples

```r
require(diptest)
x <- cbind(runif(100),runif(100))
partition <- 1+(x[1]<0.5)
d1 <- diptest.multi(x,partition)
d2 <- diptest.multi(x,partition,pvalue="tantrum",M=10)
```

Discriminant coordinates/canonical variates

Description

Computes discriminant coordinates, sometimes referred to as "canonical variates" as described in Seber (1984).

Usage

discrcoord(xd, clvecd, pool = "n", ...)

Arguments

- `xd`: the data matrix; a numerical object which can be coerced to a matrix.
- `clvecd`: integer vector of class numbers; length must equal nrow(xd).
- `pool`: string. Determines how the within classes covariance is pooled. "n" means that the class covariances are weighted corresponding to the number of points in each class (default). "equal" means that all classes get equal weight.
- `...`: no effect

Details

The matrix T (see Seber (1984), p. 270) is inverted by use of tdecomp, which can be expected to give reasonable results for singular within-class covariance matrices.
discrete.recode

Value

List with the following components

- **ev**: eigenvalues in descending order.
- **units**: columns are coordinates of projection basis vectors. New points \( x \) can be projected onto the projection basis vectors by \( x \%*% \text{units} \).
- **proj**: projections of \( xd \) onto \( \text{units} \).

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche](http://www.homepages.ucl.ac.uk/~ucakche)

References


See Also

- `plotcluster` for straight forward discriminant plots.
- `batcoord` for discriminating projections for two classes, so that also the differences in variance are shown (`discrcoord` is based only on differences in mean).
- `rFace` for generation of the example data used below.

Examples

```r
set.seed(4634)
face <- rFace(600,dMoNo=2,dNoEy=0)
gface <- as.integer(attr(face,"grouping"))
dcf <- discrcoord(face,gface)
plot(dcf$proj,col=gface)
# ...done in one step by function plotcluster.
```

---

discrete.recode  
*Recodes mixed variables dataset*

Description

Recodes a dataset with mixed continuous and categorical variables so that the continuous variables come first and the categorical variables have standard coding 1, 2, 3,... (in lexicographical ordering of values coerced to strings).

Usage

`discrete.recode(x,xvarsorted=TRUE,continuous=0,discrete)`
Arguments

x  
data matrix or data frame. The data need to be organised case-wise, i.e., if there are categorical variables only, and 15 cases with values c(1,1,2) on the 3 variables, the data matrix needs 15 rows with values 1 1 2. (Categorical variables could take numbers or strings or anything that can be coerced to factor levels as values.)

xvarssorted  
logical. If TRUE, the continuous variables are assumed to be the first ones, and the categorical variables to be behind them.

continuous  
vector of integers giving positions of the continuous variables. If xvarssorted=TRUE, a single integer, number of continuous variables.

discrete  
vector of integers giving positions of the categorical variables (the variables need to be coded in such a way that data.matrix converts them to something numeric). If xvarssorted=TRUE, a single integer, number of categorical variables.

Value

A list with components

data  
data matrix with continuous variables first and categorical variables in standard coding behind them.

ppdim  
vector of categorical variable-wise numbers of categories.

discretelevels  
list of levels of the categorical variables belonging to what is treated by flexmixedruns as category 1, 2, 3 etc.

continuous  
number of continuous variables.

discrete  
number of categorical variables.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche

See Also

lcmixed

Examples

set.seed(776655)
v1 <- rnorm(20)
v2 <- rnorm(20)
d1 <- sample(c(2,4,6,8),20,replace=TRUE)
d2 <- sample(1:4,20,replace=TRUE)
data <- cbind(v1,d1,v2,d2)
lc <-
discrete.recode(data,xvarssorted=FALSE,continuous=c(1,3),discrete=c(2,4))
require(MASS)
data(Cars93)
Cars934 <- Cars93[,c(3,5,8,10)]
cc <- discrete.recode(Cars934,xvarssorted=FALSE,continuous=c(2,3),discrete=c(1,4))
Description

An interface for ten methods of linear dimension reduction in order to separate the groups optimally in the projected data. Includes classical discriminant coordinates, methods to project differences in mean and covariance structure, asymmetric methods (separation of a homogeneous class from a heterogeneous one), local neighborhood-based methods and methods based on robust covariance matrices.

Usage

discrproj(x, clvecd, method="dc", clnum=NULL, ignorepoints=FALSE, ignorenum=0, ...)

Arguments

- **x**: the data matrix; a numerical object which can be coerced to a matrix.
- **clvecd**: vector of class numbers which can be coerced into integers; length must equal nrow(x).
- **method**: one of
  - "dc": usual discriminant coordinates, see discrcoord,
  - "bc": Bhattacharyya coordinates, first coordinate showing mean differences, second showing covariance matrix differences, see batcoord,
  - "vbc": variance dominated Bhattacharyya coordinates, see batcoord,
  - "mvdc": added mean and variance differences optimizing coordinates, see mvdcoord,
  - "adc": asymmetric discriminant coordinates, see adcoord,
  - "awc": asymmetric discriminant coordinates with weighted observations, see awcoord,
  - "arc": asymmetric discriminant coordinates with weighted observations and robust MCD-covariance matrix, see awcoord,
  - "nc": neighborhood based coordinates, see ncoord,
  - "wnc": neighborhood based coordinates with weighted neighborhoods, see ncoord,
  - "anc": asymmetric neighborhood based coordinates, see ancoord.

Note that "bc", "vbc", "adc", "awc", "arc" and "anc" assume that there are only two classes.

- **clnum**: integer. Number of the class which is attempted to plot homogeneously by "asymmetric methods", which are the methods assuming that there are only two classes, as indicated above.

- **ignorepoints**: logical. If TRUE, points with label ignorenum in clvecd are ignored in the computation for method and are only projected afterwards onto the resulting units. If pch=NULL, the plot symbol for these points is "N".
ignorenum  one of the potential values of the components of clvecd. Only has effect if ignorepoints=TRUE, see above.

...  additional parameters passed to the projection methods.

Value

discrproj returns the output of the chosen projection method, which is a list with at least the components ev, units, proj. For detailed informations see the help pages of the projection methods.

ev  eigenvalues in descending order, usually indicating portion of information in the corresponding direction.

units  columns are coordinates of projection basis vectors. New points x can be projected onto the projection basis vectors by x %*% units

proj  projections of xd onto units.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


See Also

discrcoord, batcoord, mvdcoord, adcoord, awcoord, ncoord, ancoord.

rFace for generation of the example data used below.

Examples

set.seed(4634)
face <- rFace(300,dMoNo=2,dNoEy=0,p=3)
grface <- as.integer(attr(face,"grouping"))

# The abs in the following is there to unify the output,
# because eigenvectors are defined only up to their sign.
# Statistically it doesn't make sense to compute absolute values.
round(abs(discrproj(face,grface, method="nc")$units),digits=2)
round(abs(discrproj(face,grface, method="wnc")$units),digits=2)
round(abs(discrproj(face,grface, clnum=1, method="arc")$units),digits=2)
distancefactor

Factor for dissimilarity of mixed type data

Description

Computes a factor that can be used to standardise ordinal categorical variables and binary dummy variables coding categories of nominal scaled variables for Euclidean dissimilarity computation in mixed type data. See Hennig and Liao (2013).

Usage

distancefactor(cat, n=NULL, catsizes=NULL, type="categorical", normfactor=2, qfactor=ifelse(type="categorical", 1/2, 1/(1+1/(cat-1))))

Arguments

cat integer. Number of categories of the variable to be standardised. Note that for type="categorical" the number of categories of the original variable is required, although the distancefactor is used to standardise dummy variables for the categories.

n integer. Number of data points.

catsizes vector of integers giving numbers of observations per category. One of n and catsizes must be supplied. If catsizes=NULL, rep(round(n/cat),cat) is used (this may be appropriate as well if numbers of observations of categories are unequal, if the researcher decides that the dissimilarity measure should not be influenced by empirical category sizes.

type "categorical" if the factor is used for dummy variables belonging to a nominal variable, "ordinal" if the factor is used for an ordinal variable ind standard Likert coding.

normfactor numeric. Factor on which standardisation is based. As a default, this is \( E(X_1-X_2)^2 = 2 \) for independent unit variance variables.

qfactor numeric. Factor q in Hennig and Liao (2013) to adjust for clumping effects due to discreteness.

Value

A factor by which to multiply the variable in order to make it comparable to a unit variance continuous variable when aggregated in Euclidean fashion for dissimilarity computation, so that expected effective difference between two realisations of the variable equals qfactor*normfactor.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche
References


See Also

lcmixed, pam

Examples

```r
set.seed(776655)
d1 <- sample(1:5,20,replace=TRUE)
d2 <- sample(1:4,20,replace=TRUE)
data <- cbind(d1,d2)
lc <- cat2bin(data,categorical=1)$data
lc[,1:5] <- lc[,1:5]*distancefactor(5,20,type="categorical")
lc[,6] <- lc[,6]*distancefactor(4,20,type="ordinal")
```

distcritmulti  Distance based validity criteria for large data sets

Description

Approximates average silhouette width or the Pearson version of Hubert's gamma criterion by hacking the dataset into pieces and averaging the subset-wise values, see Hennig and Liao (2013).

Usage

```r
distcritmulti(x,clustering,part=NULL,ns=10,criterion="asw", fun="dist",metric="euclidean", count=FALSE,seed=NULL,...)
```

Arguments

- `x` cases times variables data matrix.
- `clustering` vector of integers indicating the clustering.
- `part` vector of integer subset sizes; sum should be smaller or equal to the number of cases of `x`. If NULL, subset sizes are chosen approximately equal.
- `ns` integer. Number of subsets, only used if `part` is NULL.
- `criterion` "asw" or "pearson gamma", specifies whether the average silhouette width or the Pearson version of Hubert's gamma is computed.
- `fun` "dist" or "daisy", specifies which function is used for computing dissimilarities.
- `metric` passed on to `dist` (as argument method) or `daisy` to determine which dissimilarity is used.
package {distcritmulti

  count logical. if TRUE, the subset number just processed is printed.
  seed integer, random seed. (If NULL, result depends on random numbers.)
  ... further arguments to be passed on to dist or daisy.

Value

A list with components crit.overall,crit.sub,crit.sd,part.

  crit.overall value of criterion.
  crit.sub vector of subset-wise criterion values.
  crit.sd standard deviation of crit.sub, can be used to assess stability.
  subsets list of case indexes in subsets.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche

References


See Also

  cluster.stats, silhouette

Examples

  set.seed(20000)
  options(digits=3)
  face <- rFace(50, dMoNo=2, dNoEy=0, p=2)
  clustering <- as.integer(attr(face,"grouping"))
  distcritmulti(face,clustering,ns=3,seed=100000,criterion="pearsongamma")
**dridgeline**

**Density along the ridgeline**

**Description**

Computes the density of a two-component Gaussian mixture along the ridgeline (Ray and Lindsay, 2005), along which all its density extrema are located.

**Usage**

```r
dridgeline(alpha=seq(0,1,0.001), prop, 
        mu1, mu2, Sigma1, Sigma2, showplot=FALSE, ...)
```

**Arguments**

- `alpha`: sequence of values between 0 and 1 for which the density is computed.
- `prop`: mixture proportion of first component.
- `mu1`: mean vector of component 1.
- `mu2`: mean vector of component 2.
- `Sigma1`: covariance matrix of component 1.
- `Sigma2`: covariance matrix of component 2.
- `showplot`: logical. If TRUE, the density is plotted against `alpha`.
- `...`: further arguments to be passed on to plot.

**Value**

Vector of density values for values of `alpha`.

**Author(s)**

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

**References**


**Examples**

```r
q <- dridgeline(seq(0,1,0.1),0.5,c(1,1),c(2,5),diag(2),diag(2))
```
**Description**

Duda-Hart test for whether a data set should be split into two clusters.

**Usage**

```r
dudahart2(x, clustering, alpha = 0.001)
```

**Arguments**

- `x` data matrix or data frame.
- `clustering` vector of integers. Clustering into two clusters.
- `alpha` numeric between 0 and 1. Significance level (recommended to be small if this is used for estimating the number of clusters).

**Value**

A list with components

- `p.value` p-value against null hypothesis of homogeneity.
- `dh` ratio of within-cluster sum of squares for two clusters and overall sum of squares.
- `compare` critical value for `dh` at level `alpha`.
- `cluster1` FALSE if the null hypothesis of homogeneity is rejected.
- `alpha` see above.
- `z` 1-`alpha`-quantile of a standard Gaussian.

**Author(s)**

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche](http://www.homepages.ucl.ac.uk/~ucakche)

**References**


**See Also**

- `cluster.stats`

**Examples**

```r
options(digits = 2)
set.seed(98765)
iriss <- iris[sample(150, 20), -5]
km <- kmeans(iriss, 2)
dudahart2(iriss, km$cluster)
```
extract.mixturepars  

Extracts parameters for certain components from mclust

Description

Extracts parameter of certain mixture components from the output of summary.mclustBIC and updates proportions so that they sum up to 1.

Usage

extract.mixturepars(mclustsum, compnumbers, noise=FALSE)

Arguments

mclustsum  output object of summary.mclustBIC.
compnumbers  vector of integers. Numbers of mixture components.
noise  logical. Should be TRUE if a noise component was fitted by mclustBIC.

Value

Object as component parameters of summary.mclustBIC-output, but for specified components only. (Orientation information from all components is kept.)

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

Examples

set.seed(98765)
options(digits=2)
require(mclust)
iriss <- iris[sample(150,20),-5]
irisBIC <- mclustBIC(iriss)
siris <- summary(irisBIC,iriss)
exttract.mixturepars(siris,2)
findrep  

Finding representatives for cluster border

Description

Finds representative objects for the border of a cluster and the within-cluster variance as defined in the framework of the cdbw cluster validation index (and meant to be used in that context).

Usage

findrep(x, xcen, clustering, cluster, r, p = ncol(x), n = nrow(x),
        nc = sum(clustering == cluster))

Arguments

- **x**: matrix. Euclidean dataset.
- **xcen**: mean vector of cluster.
- **clustering**: vector of integers with length = nrow(x); indicating the cluster for each observation.
- **cluster**: integer. Number of cluster to be treated.
- **r**: integer. Number of representatives.
- **p**: integer. Number of dimensions.
- **n**: integer. Number of observations.
- **nc**: integer. Number of observations in cluster.

Value

List with components

- **repc**: vector of index of representatives (out of all observations).
- **repx**: vector of index of representatives (out of only the observations in cluster).
- **maxr**: number of representatives (this can be smaller than r if fewer pairwise different observations are in cluster).
- **wvar**: estimated average within-cluster squared distance to mean.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

References


See Also
cdbw

Examples

options(digits=3)
iriss <- as.matrix(iris[,c(1:5,51:55,101:105),-5])
irisc <- as.numeric(iris[,c(1:5,51:55,101:105),5])
findrep(iriss,colMeans(iriss),irisc,cluster=1,r=2)

Description

Computes Mahalanobis fixed point clusters (FPCs), i.e., subsets of the data, which consist exactly of
the non-outliers w.r.t. themselves, and may be interpreted as generated from a homogeneous normal
population. FPCs may overlap, are not necessarily exhausting and do not need a specification of the
number of clusters.

Note that while fixmahal has lots of parameters, only one (or few) of them have usually to be
specified, cf. the examples. The philosophy is to allow much flexibility, but to always provide
sensible defaults.

Usage

fixmahal(dat, n = nrow(as.matrix(dat)), p = ncol(as.matrix(dat)),
method = "fuzzy", cgen = "fixed",
ca = NA, ca2 = NA,
calpha = ifelse(method=="fuzzy",0.95,0.99),
calpha2 = 0.995,
pointit = TRUE, subset = n,
nc1 = 100+28*p,
startn = 18*p, mnc = floor(startn/2),
mer = ifelse(pointit,0.1,0),
distcut = 0.85, maxit = 5*n, iter = n*1e-5,
init.group = list(),
ind.storage = TRUE, countmode = 100,
plot = "none")

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

## S3 method for class 'summary.mfpc'
print(x, maxnc=30, ...)

## S3 method for class 'mfpc'
plot(x, dat, no, bw=FALSE, main=c("Representative FPC No. ",no),
  xlab=NULL, ylab=NULL,
  pch=NULL, col=NULL, ...)

## S3 method for class 'mfpc'
fpclusters(object, dat=NA, ca=object$ca, p=object$p, ...)

fpmi(dat, n = nrow(as.matrix(dat)), p = ncol(as.matrix(dat)),
  gv, ca, ca2, method = "ml", plot,
  maxit = 5*n, iter = n*1e-6)

Arguments

- **dat**: something that can be coerced to a numerical matrix or vector. Data matrix, rows are points, columns are variables. `fpclusters.rfpc` does not need specification of `dat` if `fixmahal` has been run with `ind.storage=TRUE`.

- **n**: optional positive integer. Number of cases.

- **p**: optional positive integer. Number of independent variables.

- **method**: a string. `method="classical"` means 0-1 weighting of observations by Mahalanobis distances and use of the classical normal covariance estimator. `method="ml"` uses the ML-covariance estimator (division by `n` instead of `n-1`). This is used in Hennig and Christlieb (2002). `method` can also be "mcd" or "mve", to enforce the use of robust centers and covariance matrices, see `cov.rob`. This is experimental, not recommended at the moment, may be very slowly and requires library `lqs`. The default is `method="fuzzy"`, where weighted means and covariance matrices are used (Hennig, 2005). The weights are computed by `wfu`, i.e., a function that is constant 1 for arguments smaller than `ca`, 0 for arguments larger than `ca2` and continuously linear in between. Convergence is only proven for `method="ml"` up to now.

- **cgen**: optional string. "fixed" means that the same tuning constant `ca` is used for all iterations. "auto" means that `ca` is generated dependently on the size of the current data subset in each iteration by `cmahal`. This is experimental.

- **ca**: optional positive number. Tuning constant, specifying required cluster separation. By default determined as `calpha`-quantile of the chisquared distribution with `p` degrees of freedom.

- **ca2**: optional positive number. Second tuning constant needed if `method="fuzzy"`. By default determined as `calpha2`-quantile of the chisquared distribution with `p` degrees of freedom.

- **calpha**: number between 0 and 1. See `ca`.

- **calpha2**: number between 0 and 1, larger than `calpha`. See `ca2`.

- **pointit**: optional logical. If `TRUE`, subset fixed point algorithms are started from initial configurations, which are built around single points of the dataset, cf. `mahalconf`. Otherwise, initial configurations are only specified by `init.group`.

- **subset**: optional positive integer smaller or equal than `n`. Initial configurations for the fixed point algorithm (cf. `mahalconf`) are built from a subset of `subset` points from the data. No effect if `pointit=FALSE`. Default: all points.
**nc1**
optional positive integer. Tuning constant needed by `cmahal` to generate clusters automatically. Only needed for `cgen="auto"`.

**startn**
optional positive integer. Size of the initial configurations. The default value is chosen to prevent that small meaningless FPCs are found, but it should be decreased if clusters of size smaller than the default value are of interest.

**mnc**
optional positive integer. Minimum size of clusters to be reported.

**mer**
optional nonnegative number. FPCs (groups of them, respectively, see details) are only reported as stable if the ratio of the number of their findings to their number of points exceeds `mer`. This holds under `pointit=TRUE` and `subset=n`. For `subset<n`, the ratio is adjusted, but for small `subset`, the results may extremely vary and have to be taken with care.

**distcut**
optional value between 0 and 1. A similarity measure between FPCs, given in Hennig (2002), and the corresponding Single Linkage groups of FPCs with similarity larger than `distcut` are computed. A single representative FPC is selected for each group.

**maxit**
optional integer. Maximum number of iterations per algorithm run (usually an FPC is found much earlier).

**iter**
positive number. Algorithm stops when difference between subsequent weight vectors is smaller than `iter`. Only needed for `method="fuzzy"`.

**init.group**
optional list of logical vectors of length `n`. Every vector indicates a starting configuration for the fixed point algorithm. This can be used for datasets with high dimension, where the vectors of `init.group` indicate cluster candidates found by graphical inspection or background knowledge, as in Hennig and Christlieb (2002).

**ind.storage**
optional logical. If `TRUE`, then all indicator vectors of found FPCs are given in the value of `fixmahal`. May need lots of memory, but is a bit faster.

**countmode**
optional positive integer. Every `countmode` algorithm runs `fixmahal` shows a message.

**plot**
optional string. If "start", you get a scatterplot of the first two variables to highlight the initial configuration. "iteration" generates such a plot at each iteration, "both" does both (this may be very time consuming). The default is "none".

**object**
oobject of class `mfpc`, output of `fixmahal`.

**x**
oobject of class `mfpc`, output of `fixmahal`.

**maxnc**
positive integer. Maximum number of FPCs to be reported.

**no**
positive integer. Number of the representative FPC to be plotted.

**bw**
ooptional logical. If `TRUE`, plot is black/white, FPC is indicated by different symbol. Else FPC is indicated red.

**main**
plot title.

**xlab**
label for x-axis. If NULL, a default text is used.

**ylab**
label for y-axis. If NULL, a default text is used.

**pch**
plotting symbol, see `par`. If NULL, the default is used.
Details

A (crisp) Mahalanobis FPC is a data subset that reproduces itself under the following operation:
Compute mean and covariance matrix estimator for the data subset, and compute all points of the
data set for which the squared Mahalanobis distance is smaller than \( c_a \).
Fixed points of this operation can be considered as clusters, because they contain only non-outliers
(as defined by the above mentioned procedure) and all other points are outliers w.r.t. the subset.
The current default is to compute fuzzy Mahalanobis FPCs, where the points in the subset have a
membership weight between 0 and 1 and give rise to weighted means and covariance matrices. The
new weights are then obtained by computing the weight function \( wfu \) of the squared Mahalanobis
distances, i.e., full weight for squared distances smaller than \( c_a \), zero weight for squared distances
larger than \( c_a R \) and decreasing weights (linear function of squared distances) in between.
A fixed point algorithm is started from the whole dataset, algorithms are started from the subsets
specified in \texttt{initNgroup}, and further algorithms are started from further initial configurations as
explained under subset and in the function \texttt{mahalconf}.
Usually some of the FPCs are unstable, and more than one FPC may correspond to the same significa-
ticant pattern in the data. Therefore the number of FPCs is reduced: A similarity matrix is computed
between FPCs. Similarity between sets is defined as the ratio between 2 times size of intersection
and the sum of sizes of both sets. The Single Linkage clusters (groups) of level \( \text{distcut} \) are
computed, i.e. the connectivity components of the graph where edges are drawn between FPCs
with similarity larger than \( \text{distcut} \). Groups of FPCs whose members are found often enough (cf.
parameter \( \text{mer} \)) are considered as stable enough. A representative FPC is chosen for every Single
Linkage cluster of FPCs according to the maximum expectation ratio \( \text{ser} \). \( \text{ser} \) is the ratio be-
tween the number of findings of an FPC and the number of points of an FPC, adjusted suitably if
\( \text{subset}<\text{n} \). Usually only the representative FPCs of stable groups are of interest.
Default tuning constants are taken from Hennig (2005).
Generally, the default settings are recommended for \texttt{fixmahal}. For large datasets, the use of
\texttt{initNgroup} together with \texttt{pointit=FALSE} is useful. Occasionally, \texttt{mnc} and \texttt{startn}
may be chosen smaller than the default, if smaller clusters are of interest, but this may lead to too many clusters.
Decrease of \( c_a \) will often lead to too many clusters, even for homogeneous data. Increase of \( c_a \) will
produce only very strongly separated clusters. Both may be of interest occasionally.
Singlar covariance matrices during the iterations are handled by \texttt{solvecov}.

Value

\texttt{fixmahal} returns an object of class \texttt{mfpc}. This is a list containing the components \texttt{nc}, \texttt{g}, \texttt{means}, \texttt{covs}, \texttt{nfound}, \texttt{er}, \texttt{tsc},

summary.mfpc returns an object of class summary.mfpc. This is a list containing the components:

- `means`: list of numerical vectors. Means of FPCs. In `summary.mfpc`, only for representative FPCs of stable groups and sorted according to `ser`.
- `covs`: list of numerical matrices. Covariance matrices of FPCs. In `summary.mfpc`, only for representative FPCs of stable groups and sorted according to `ser`.
- `nfound`: vector of integers. Number of findings for the FPCs.
- `er`: numerical vector. Ratio of number of findings of FPCs to their size. Under `pointit=TRUE`, this can be taken as a measure of stability of FPCs.
- `tsc`: integer. Number of algorithm runs leading to too small or too seldom found FPCs.
- `ncoll`: integer. Number of algorithm runs where collinear covariance matrices occurred.
- `skc`: integer. Number of skipped clusters.
- `grto`: vector of integers. Numbers of FPCs to which algorithm runs led, which were started by `init.group`.
- `imatrix`: vector of integers. Size of intersection between FPCs. See `sseg`.
- `smatrix`: numerical vector. Similarities between FPCs. See `sseg`.
- `stn`: integer. Number of representative FPCs of stable groups. In `summary.mfpc`, sorted according to `ser`.
- `stfound`: vector of integers. Number of findings of members of all groups of FPCs. In `summary.mfpc`, sorted according to `ser`.
- `ser`: numerical vector. Ratio of number of findings of groups of FPCs to their size. Under `pointit=TRUE`, this can be taken as a measure of stability of FPCs. In `summary.mfpc`, sorted from largest to smallest.
- `sfpc`: vector of integers. Numbers of representative FPCs of all groups.
- `ssig`: vector of integers of length `stn`. Numbers of representative FPCs of the stable groups.
- `sto`: vector of integers. Numbers of groups ordered according to largest `ser`.
- `struc`: vector of integers. Number of group an FPC belongs to.
- `n`: see arguments.
- `p`: see arguments.
- `method`: see arguments.
- `cgen`: see arguments.
- `ca`: see arguments, if `cgen` has been "fixed". Else numerical vector of length `nc` (see below), giving the final values of `ca` for all FPC. In `fpmi`, tuning constant for the iterated FPC.

fpclusters.mfpc returns a list of indicator vectors for the representative FPCs of stable groups.
fpmi returns a list with the components `mg`, `covg`, `md`, `gv`, `coll`, `method`, `ca`.

nc
integer. Number of FPCs.

g
list of logical vectors. Indicator vectors of FPCs. FALSE if ind. storage=FALSE.

means
list of numerical vectors. Means of FPCs. In `summary.mfpc`, only for representative FPCs of stable groups and sorted according to `ser`.

covs
list of numerical matrices. Covariance matrices of FPCs. In `summary.mfpc`, only for representative FPCs of stable groups and sorted according to `ser`.

nfound
vector of integers. Number of findings for the FPCs.

er
numerical vector. Ratio of number of findings of FPCs to their size. Under `pointit=TRUE`, this can be taken as a measure of stability of FPCs.

tsc
integer. Number of algorithm runs leading to too small or too seldom found FPCs.

ncoll
integer. Number of algorithm runs where collinear covariance matrices occurred.

skc
integer. Number of skipped clusters.

gtore
vector of integers. Numbers of FPCs to which algorithm runs led, which were started by `init.group`.

imatrix
vector of integers. Size of intersection between FPCs. See `sseg`.

smatrix
numerical vector. Similarities between FPCs. See `sseg`.

stn
integer. Number of representative FPCs of stable groups. In `summary.mfpc`, sorted according to `ser`.

stfound
vector of integers. Number of findings of members of all groups of FPCs. In `summary.mfpc`, sorted according to `ser`.

ser
numerical vector. Ratio of number of findings of groups of FPCs to their size. Under `pointit=TRUE`, this can be taken as a measure of stability of FPCs. In `summary.mfpc`, sorted from largest to smallest.

sfpc
vector of integers. Numbers of representative FPCs of all groups.

ssig
vector of integers of length `stn`. Numbers of representative FPCs of the stable groups.

sto
vector of integers. Numbers of groups ordered according to largest `ser`.

struc
vector of integers. Number of group an FPC belongs to.

n
see arguments.

p
see arguments.

method
see arguments.

cgen
see arguments.

ca
see arguments, if `cgen` has been "fixed". Else numerical vector of length `nc` (see below), giving the final values of `ca` for all FPC. In `fpmi`, tuning constant for the iterated FPC.
fixmahal

ca2 see arguments.
cvec numerical vector of length n for cgen="auto". The values for the tuning constant ca corresponding to the cluster sizes from 1 to n.
calpha see arguments.
pointit see arguments.
subset see arguments.
mnc see arguments.
startn see arguments.
mer see arguments.
distcut see arguments.

sn vector of integers. Number of points of representative FPCs.
tskip integer. Number of algorithm runs leading to skipped FPCs.
sim vector of integers. Size of intersections between representative FPCs of stable groups. See sseg.
mg mean vector.
covg covariance matrix.
md Mahalanobis distances.
gv logical (numerical, respectively, if method="fuzzy") indicator vector of iterated FPC.
coll logical. TRUE means that singular covariance matrices occurred during the iterations.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


See Also

fixreg for linear regression fixed point clusters.
mahalconf, wfu, cmahal for computation of initial configurations, weights, tuning constants.
sseg for indexing the similarity/intersection vectors computed by fixmahal.
batcoord, cov.rob, solvecov, cov.wml, plotcluster for computation of projections, (inverted) covariance matrices, plotting.
rFace for generation of example data, see below.
Examples

options(digits=2)
set.seed(20000)
face <- rFace(400,dMoNo=2,dNoEy=0, p=3)
# The first example uses grouping information via init.group.
initg <- list()
grface <- as.integer(attr(face,"grouping"))
for (i in 1:5) initg[[i]] <- (grface==i)
ff0 <- fixmahal(face, pointit=FALSE, init.group=initg)
summary(ff0)
cff0 <- fpclusters(ff0)
plot(face, col=1+cff0[[1]])
plot(face, col=1+cff0[[4]]) # Why does this come out as a cluster?
plot(ff0, face, 4) # A bit clearer...
# Without grouping information, examples need more time:
# ff1 <- fixmahal(face)
# summary(ff1)
# cff1 <- fpclusters(ff1)
# plot(face, col=1+cff1[[1]])
# plot(face, col=1+cff1[[4]]) # Why does this come out as a cluster?
# plot(ff1, face, 6) # A bit clearer...
# ff2 <- fixmahal(face,method="ml")
# summary(ff2)
# ff3 <- fixmahal(face,method="ml",calpha=0.95, subset=50)
# summary(ff3)
## ...fast, but lots of clusters. mer=0.3 may be useful here.
# set.seed(30000)
# face2 <- rFace(400,dMoNo=2,dNoEy=0)
# ff5 <- fixmahal(face2)
# summary(ff5)
## misses right eye of face data; with p=6,
## initial configurations are too large for 40 point clusters
# ff6 <- fixmahal(face2, startn=30)
# summary(ff6)
# cff6 <- fpclusters(ff6)
# plot(face2, col=1+cff6[[3]])
# plot(ff6, face2, 3)
# x <- c(1,2,3,6,6,7,8,120)
# ff8 <- fixmahal(x)
# summary(ff8)
##...dataset a bit too small for the defaults...
# ff9 <- fixmahal(x, mnc=3, startn=3)
# summary(ff9)

fixreg

Linear Regression Fixed Point Clusters

Description

Computes linear regression fixed point clusters (FPCs), i.e., subsets of the data, which consist exactly of the non-outliers w.r.t. themselves, and may be interpreted as generated from a homogeneous
linear regression relation between independent and dependent variable. FPCs may overlap, are not necessarily exhausting and do not need a specification of the number of clusters.

Note that while fixreg has lots of parameters, only one (or few) of them have usually to be specified, cf. the examples. The philosophy is to allow much flexibility, but to always provide sensible defaults.

Usage

```r
fixreg(indep, dep, n=length(dep),
       p=ncol(as.matrix(indep)),
       ca=NA, mnc=NA, mtf=3, ir=NA, irnc=NA,
       irprob=0.95, mncprob=0.5, maxir=20000, maxit=5*n,
       distcut=0.85, init.group=list(),
       ind.storage=FALSE, countmode=100,
       plot=FALSE)
```

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

`# S3 method for class 'summary.rfpc'
print(x, maxnc=30, ...)`

`# S3 method for class 'rfpc'
plot(x, indep, dep, no, bw=TRUE,
     main=c("Representative FPC No. ",no),
     xlab="Linear combination of independents",
     ylab=deparse(substitute(indep)),
     xlim=NULL, ylim=range(dep),
     pch=NULL, col=NULL,...)`

`# S3 method for class 'rfpc'
fpclusters(object, indep, dep, ca, ...)`

`rfpi(indep, dep, p, gv, ca, maxit, plot)`

Arguments

- `indep` numerical matrix or vector. Independent variables. Leave out for clustering one-dimensional data. `fpclusters.rfpc` does not need specification of `indep` if `fixreg` was run with `ind.storage=TRUE`.
- `dep` numerical vector. Dependent variable. `fpclusters.rfpc` does not need specification of `dep` if `fixreg` was run with `ind.storage=TRUE`.
- `n` optional positive integer. Number of cases.
- `p` optional positive integer. Number of independent variables.
- `ca` optional positive number. Tuning constant, specifying required cluster separation. By default determined automatically as a function of `n` and `p`, see function `can`, Hennig (2002a).
**mnc**
optional positive integer. Minimum size of clusters to be reported. By default determined automatically as a function of \( mncprob \). See Hennig (2002a).

**mtf**
optional positive integer. FPCs must be found at least \( mtf \) times to be reported by `summary.rfpc`.

**ir**
optional positive integer. Number of algorithm runs. By default determined automatically as a function of \( n, p, irnc, irprob, mtf, maxir \). See function `itnumber` and Hennig (2002a).

**irnc**
optional positive integer. Size of the smallest cluster to be found with approximated probability \( irprob \).

**irprob**
optional value between 0 and 1. Approximated probability for a cluster of size \( irnc \) to be found.

**mncprob**
optional value between 0 and 1. Approximated probability for a cluster of size \( mnc \) to be found.

**maxir**
optional integer. Maximum number of algorithm runs.

**maxit**
optional integer. Maximum number of iterations per algorithm run (usually an FPC is found much earlier).

**distcut**
optional value between 0 and 1. A similarity measure between FPCs, given in Hennig (2002a), and the corresponding Single Linkage groups of FPCs with similarity larger than \( distcut \) are computed. A single representative FPC is selected for each group.

**init.group**
optional list of logical vectors of length \( n \). Every vector indicates a starting configuration for the fixed point algorithm. This can be used for datasets with high dimension, where the vectors of `init.group` indicate cluster candidates found by graphical inspection or background knowledge.

**ind.storage**
optional logical. If `TRUE`, then all indicator vectors of found FPCs are given in the value of `fixreg`. May need lots of memory, but is a bit faster.

**countmode**
optional positive integer. Every `countmode` algorithm runs `fixreg` shows a message.

**plot**
optional logical. If `TRUE`, you get a scatterplot of first independent vs. dependent variable at each iteration.

**object**
object of class `rfpc`, output of `fixreg`.

**x**
object of class `rfpc`, output of `fixreg`.

**maxnc**
positive integer. Maximum number of FPCs to be reported.

**no**
positive integer. Number of the representative FPC to be plotted.

**bw**
optional logical. If `TRUE`, plot is black/white, FPC is indicated by different symbol. Else FPC is indicated red.

**main**
plot title.

**xlab**
label for x-axis.

**ylab**
label for y-axis.

**xlim**
plotted range of x-axis. If `NULL`, the range of the plotted linear combination of independent variables is used.

**ylim**
plotted range of y-axis.
A linear regression FPC is a data subset that reproduces itself under the following operation: Compute linear regression and error variance estimator for the data subset, and compute all points of the dataset for which the squared residual is smaller than \( c a \) times the error variance. Fixed points of this operation can be considered as clusters, because they contain only non-outliers (as defined by the above mentioned procedure) and all other points are outliers w.r.t. the subset. `fixreg` performs \( ir \) fixed point algorithms started from random subsets of size \( p+R \) to look for FPCs. Additionally an algorithm is started from the whole dataset, and algorithms are started from the subsets specified in `initNgroup`. Usually some of the FPCs are unstable, and more than one FPC may correspond to the same significant pattern in the data. Therefore the number of FPCs is reduced: FPCs with less than \( m nc \) points are ignored. Then a similarity matrix is computed between the remaining FPCs. Similarity between sets is defined as the ratio between 2 times size of intersection and the sum of sizes of both sets. The Single Linkage clusters (groups) of level \( distcut \) are computed, i.e. the connectivity components of the graph where edges are drawn between FPCs with similarity larger than \( distcut \). Groups of FPCs whose members are found \( m tf \) times or more are considered as stable enough. A representative FPC is chosen for every Single Linkage cluster of FPCs according to the maximum expectation ratio \( ser \). \( ser \) is the ratio between the number of findings of an FPC and the estimated expectation of the number of findings of an FPC of this size, called expectation ratio and computed by `clusexpect`. Usually only the representative FPCs of stable groups are of interest. The choice of the involved tuning constants such as \( c a \) and \( ir \) is discussed in detail in Hennig (2002a). Statistical theory is presented in Hennig (2003). Generally, the default settings are recommended for `fixreg`. In cases where they lead to a too large number of algorithm runs (e.g., always for \( p>4 \)), the use of `initNgroup` together with `mtf=1` and `ir=0` is useful. Occasionally, `irnc` may be chosen smaller than the default, if smaller clusters are of interest, but this may lead to too many clusters and too many algorithm runs. Decrease of \( c a \) will often lead to too many clusters, even for homogeneous data. Increase of \( c a \) will produce only very strongly separated clusters. Both may be of interest occasionally.  

\( rfpi \) is called by `fixreg` for a single fixed point algorithm and will usually not be executed alone. \( summary.rfpc \) gives a summary about the representative FPCs of stable groups.  

\( plot.rfpc \) is a plot method for the representative FPC of stable group no. no. It produces a scatter-plot of the linear combination of independent variables determined by the regression coefficients of the FPC vs. the dependent variable. The regression line and the region of non-outliers determined by \( c a \) are plotted as well.  

\( fpclusters.rfpc \) produces a list of indicator vectors for the representative FPCs of stable groups.  

**Value**  
`fixreg` returns an object of class `rfpc`. This is a list containing the components `nc`, `g`, `coefs`, `vars`, `nfound`, `er`, `tsc`, etc.
summary.rfpc returns an object of class summary.rfpc. This is a list containing the components coefs, vars, stfound, stn, sn, ser, tsc, sim, ca, ir, mnc, mtf.
fpclusters.rfpc returns a list of indicator vectors for the representative FPCs of stable groups.
rftp.i returns a list with the components coef, var, g, coll, ca.

nc integer. Number of FPCs.
g list of logical vectors. Indicator vectors of FPCs. FALSE if ind.Nstorage=FALSE.
coefs list of numerical vectors. Regression coefficients of FPCs. In summary.rfpc, only for representative FPCs of stable groups and sorted according to stfound.
vars list of numbers. Error variances of FPCs. In summary.rfpc, only for representative FPCs of stable groups and sorted according to stfound.
nfound vector of integers. Number of findings for the FPCs.
er numerical vector. Expectation ratios of FPCs. Can be taken as a stability measure.
tsc integer. Number of algorithm runs leading to too small or too seldom found FPCs.
ncoll integer. Number of algorithm runs where collinear regressor matrices occurred.
grto vector of integers. Numbers of FPCs to which algorithm runs led, which were started by init.group.
imatrix vector of integers. Size of intersection between FPCs. See sseg.
smatrix numerical vector. Similarities between FPCs. See sseg.
sto vector of integers. Number of group an FPC belongs to.
struc vector of integers. Number of points of representative FPCs.
n see arguments.
p see arguments.
ca see arguments.
ir see arguments.
mnc see arguments.
mtf see arguments.
distcut see arguments.
ssig vector of integers. As sfpc, but only for stable groups.
sto vector of integers. Number of representative FPC of most, 2nd most, ..., often found group of FPCs.

sim vector of integers. Size of intersections between representative FPCs of stable groups. See sseg.

coef vector of regression coefficients.

var error variance.

g logical indicator vector of iterated FPC.

coll logical. TRUE means that singular covariance matrices occurred during the iterations.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


See Also

fixmahal for fixed point clusters in the usual setup (non-regression).

regmix for clusterwise linear regression by mixture modeling ML.

can, itnumber for computation of the default settings.

clusexpect for estimation of the expected number of findings of an FPC of given size.

itnumber for the generation of the number of fixed point algorithms.

minsize for the smallest FPC size to be found with a given probability.

sseg for indexing the similarity/intersection vectors computed by fixreg.

Examples

set.seed(190000)
options(digits=3)
data(tonedata)
attach(tonedata)
tonefix <- fixreg(stretchratio,tuned,mtf=1,ir=20)
summary(tonefix)
  # This is designed to have a fast example; default setting would be better.
  # If you want to see more (and you have a bit more time),
  # try out the following:
  ## Not run:
  set.seed(1000)
tonefix <- fixreg(stretchratio,tuned)
  # Default – good for these data
  summary(tonefix)
  plot(tonefix,stretchratio,tuned,1)
  plot(tonefix,stretchratio,tuned,2)
plot(tonefix, stretchratio, tuned, 3, bw=FALSE, pch=5)
toneclus <- fpclusters(tonefix, stretchratio, tuned)
plot(stretchratio, tuned, col=1+toneclus[[2]])
tonefix2 <- fixreg(stretchratio, tuned, distcut=1, mtf=1, countmode=50)
# Every found fixed point cluster is reported,
# no matter how unstable it may be.
summary(tonefix2)
tonefix3 <- fixreg(stretchratio, tuned, ca=7)
# ca defaults to 10.87 for these data.
summary(tonefix3)
subset <- c(rep(FALSE, 5), rep(TRUE, 24), rep(FALSE, 12))
tonefix4 <- fixreg(stretchratio, tuned, 
                   mtf=1, ir=0, init.group=list(subset))
summary(tonefix4)

## End(Not run)

---

flexmixedruns Fitting mixed Gaussian/multinomial mixtures with flexmix

Description

flexmixedruns fits a latent class mixture (clustering) model where some variables are continuous and modelled within the mixture components by Gaussian distributions and some variables are categorical and modelled within components by independent multinomial distributions. The fit is by maximum likelihood estimation computed with the EM-algorithm. The number of components can be estimated by the BIC.

Note that at least one categorical variable is needed, but it is possible to use data without continuous variable.

Usage

flexmixedruns(x, diagonal=TRUE, xvarssorted=TRUE, 
              continuous, discrete, ppdim=NULL, initial.cluster=NULL, 
              simruns=20, n.cluster=1:20, verbose=TRUE, recode=TRUE, 
              allout=TRUE, control=list(minprior=0.001), silent=TRUE)

Arguments

x data matrix or data frame. The data need to be organised case-wise, i.e., if there are categorical variables only, and 15 cases with values c(1,1,2) on the 3 variables, the data matrix needs 15 rows with values 1 1 2. (Categorical variables could take numbers or strings or anything that can be coerced to factor levels as values.)

diagonal logical. If TRUE, Gaussian models are fitted restricted to diagonal covariance matrices. Otherwise, covariance matrices are unrestricted. TRUE is consistent with the “within class independence” assumption for the multinomial variables.
flexmixedruns

xvars Sorted logical. If TRUE, the continuous variables are assumed to be the first ones, and the categorical variables to be behind them.

Continuous vector of integers giving positions of the continuous variables. If xvars Sorted=TRUE, a single integer, number of continuous variables.

Discrete vector of integers giving positions of the categorical variables. If xvars Sorted=TRUE, a single integer, number of categorical variables.

Ppdim vector of integers specifying the number of (in the data) existing categories for each categorical variable. If recode=TRUE, this can be omitted and is computed automatically.

Initial.cluster this corresponds to the cluster parameter in flexmix and should only be specified if simruns=1 and n.cluster is a single number. Either a matrix with n.cluster columns of initial cluster membership probabilities for each observation; or a factor or integer vector with the initial cluster assignments of observations at the start of the EM algorithm. Default is random assignment into n.cluster clusters.

Simruns integer. Number of starts of the EM algorithm with random initialisation in order to find a good global optimum.

N.cluster vector of integers, numbers of components (the optimum one is found by minimising the BIC).

Verbose logical. If TRUE, some information about the different runs of the EM algorithm is given out.

Recode logical. If TRUE, the function discrete.recode is applied in order to recode categorical data so that the lcmixed-method can use it. Only set this to FALSE if your data already has that format (even in that case, TRUE doesn't do harm). If recode=FALSE, the categorical variables are assumed to be coded 1,2,3,...

Allout logical. If TRUE, the regular flexmix-output is given out for every single number of clusters, which can create a huge output object.

Control list of control parameters for flexmix, for details see the help page of FLXcontrol-class.

Silent logical. This is passed on to the try-function. If FALSE, error messages from failed runs of flexmix are suppressed. (The information that a flexmix-error occurred is still given out if verbose=TRUE).

Details

Sometimes flexmix produces errors because of degenerating covariance matrices, too small clusters etc. flexmixedruns tolerates these and treats them as non-optimal runs. (Higher simruns or different control may be required to get a valid solution.)

General documentation on flexmix can be found in Friedrich Leisch's "FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R", https://CRAN.R-project.org/package=flexmix

Value

A list with components
optsummary summary object for flexmix object with optimal number of components.
optimalk optimal number of components.
errcount vector with numbers of EM runs for each number of components that led to flexmix errors.
flexout if allout=TRUE, list of flexmix output objects for all numbers of components, for details see the help page of flexmix-class. Slots that can be used include for example cluster and components. So if fo is the flexmixedruns-output object, fo$flexout[[fo$optimalk]]@cluster gives a component number vector for the observations (maximum posterior rule), and fo$flexout[[fo$optimalk]]@components gives the estimated model parameters, which for lcmixed and therefore flexmixedruns are called
center mean vector
cov covariance matrix
pp list of categorical variable-wise category probabilities
If allout=FALSE, only the flexmix output object for the optimal number of components, i.e., the [[fo$optimalk]] indexing above can then be omitted.
bicvals vector of values of the BIC for each number of components.
ppdim vector of categorical variable-wise numbers of categories.
discretelevels list of levels of the categorical variables belonging to what is treated by flexmixedruns as category 1, 2, 3 etc.

Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche

References

See Also
lcmixed, flexmix, FLXcontrol-class, flexmix-class, discrete.recode.

Examples
options(digits=3)
set.seed(776655)
v1 <- rnorm(100)
v2 <- rnorm(100)
d1 <- sample(1:5,100,replace=TRUE)
d2 <- sample(1:4,100,replace=TRUE)
data <- cbind(v1,v2,d1,d2)
fr <- flexmixedruns(data,continuous=2,discrete=2,simruns=2,n.cluster=2:3,allout=FALSE)
print(fr$optimalk)
**fpclusters**

(*)Extracting clusters from fixed point cluster objects (*)

**Description**

`fpclusters` is a generic function which extracts the representative fixed point clusters (FPCs) from FPC objects generated by `fixmahal` and `fixreg`. For documentation and examples see `fixmahal` and `fixreg`.

**Usage**

`fpclusters(object, ...)`

**Arguments**

- `object` object of class `rfpc` or `mfpc`.
- `...` further arguments depending on the method.

**Value**

a list of logical or numerical vectors indicating or giving the weights of the cluster memberships.

**Author(s)**

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

**See Also**

`fixmahal`, `fixreg`

---

**itnumber**

(*)Number of regression fixed point cluster iterations (*)

**Description**

Computes the number of fixed point iterations needed by `fixreg` to find `mtf` times a fixed point cluster (FPC) of size `cn` with an approximated probability of `prob`.

Though for use within `fixreg`.

**Usage**

`itnumber(n, p, cn, mtf, prob = 0.95, maxir = 20000)`
Arguments

n  
  positive integer. Total number of points.

p  
  positive integer. Number of independent variables.

cn  
  positive integer smaller or equal to n. Size of the FPC.

mtf  
  positive integer.

prob  
  number between 0 and 1.

maxir  
  positive integer. Itnumber is set to this value if it would otherwise be larger.

Details

The computation is based on the binomial distribution with probability given by clusexpect with ir=1.

Value

An integer.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


See Also

fixreg, clusexpect

Examples

jittervar(500,4,150,2)

jittervar  
Jitter variables in a data matrix

Description

Jitters some variables in a data matrix.

Usage

jittervar(x,jitterv=NULL,factor=1)
kmeansCBI

Arguments

- **x**: data matrix or data frame.
- **jitterv**: vector of numbers of variables to be jittered.
- **factor**: numeric. Passed on to `jitter`. See the documentation there. The higher, the more jittering.

Value

data matrix or data frame with jittered variables.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche](http://www.homepages.ucl.ac.uk/~ucakche)

See Also

- `jitter`

Examples

```r
set.seed(776655)
v1 <- rnorm(20)
v2 <- rnorm(20)
d1 <- sample(1:5,20,replace=TRUE)
d2 <- sample(1:4,20,replace=TRUE)
data <- cbind(v1,v2,d1,d2)
jv <- jittervar(data,jitterv=3:4)
```

Description

These functions provide an interface to several clustering methods implemented in R, for use together with the cluster stability assessment in `clusterboot` (as parameter `clustermethod`; "CBI" stands for "clusterboot interface"). In some situations it could make sense to use them to compute a clustering even if you don't want to run `clusterboot`, because some of the functions contain some additional features (e.g., normal mixture model based clustering of dissimilarity matrices projected into the Euclidean space by MDS or partitioning around medoids with estimated number of clusters, noise/outlier identification in hierarchical clustering).
Usage

kmeansCBI(data, krange, k, scaling=FALSE, runs=1, criterion="ch", ...)

hclustCBI(data, k, cut="number", method, scaling=TRUE, noisecut=0, ...)

hclusttreeCBI(data, minlevel=2, method, scaling=TRUE, ...)

disthclustCBI(dmatrix, k, cut="number", method, noisecut=0, ...)

noisemclustCBI(data, G, k, modelNames, nnk, hcmodel=NULL, Vinv=NULL, summary.out=FALSE, ...)

distnoisemclustCBI(dmatrix, G, k, modelNames, nnk, hcmodel=NULL, Vinv=NULL, mds.method="classical", mdsdim=4, summary.out=FALSE, points.out=FALSE, ...)

claraCBI(data, k, usepam=TRUE, diss=inherits(data,"dist"), ...)

pamkCBI(data, krange=2:10, k=NULL, criterion="asw", usepam=TRUE, scaling=TRUE, diss=inherits(data,"dist"), ...)

trimkmeansCBI(data, k, scaling=TRUE, trim=0.1, ...)

tclustCBI(data, k, trim=0.05, ...)

disttrimkmeansCBI(dmatrix, k, scaling=TRUE, trim=0.1, mds.method="classical", mdsdim=4, ...)

dbscanCBI(data, eps, MinPts, diss=inherits(data,"dist"), ...)

mahalCBI(data, clustercut=0.5, ...)

mergenormCBI(data, G=NULL, k=NULL, modelNames=NULL, nnk=0, hcmodel = NULL, Vinv = NULL, mergemethod="bhat", cutoff=0.1, ...)

speccCBI(data, k, ...)

pdfclustCBI(data, ...)

Arguments

data          a numeric matrix. The data matrix - usually a cases*variables-data matrix.
well, see parameter `diss`.

Parameter `dmatrix` is a squared numerical dissimilarity matrix or a `dist`-object.

Parameter `k` is numeric, usually integer. In most cases, this is the number of clusters for methods where this is fixed. For `hclustCBI` and `disthclustCBI` see parameter `cut` below. Some methods have a `k` parameter on top of a `G` or `krange` parameter for compatibility; `k` in these cases does not have to be specified but if it is, it is always a single number of clusters and overwrites `G` and `krange`.

Parameter `scaling` is either a logical value or a numeric vector of length equal to the number of variables. If scaling is a numeric vector with length equal to the number of variables, then each variable is divided by the corresponding value from `scaling`. If scaling is TRUE then scaling is done by dividing the (centered) variables by their root-mean-square, and if scaling is FALSE, no scaling is done before execution.

Parameter `runs` is an integer. Number of random initializations from which the k-means algorithm is started.

Parameter `criterion` is either "ch" or "asw". Decides whether number of clusters is estimated by the Calinski-Harabasz criterion or by the average silhouette width.

Parameter `cut` is either "level" or "number". This determines how `cutree` is used to obtain a partition from a hierarchy tree. `cut="level"` means that the tree is cut at a particular dissimilarity level, `cut="number"` means that the tree is cut in order to obtain a fixed number of clusters. The parameter `k` specifies the number of clusters or the dissimilarity level, depending on `cut`.

Parameter `method` is a method for hierarchical clustering, see the documentation of `hclust`.

Parameter `noisecut` is numeric. All clusters of size \(< noisecut \) in the `disthclustCBI/hclustCBI`-partition are joined and declared as noise/outliers.

Parameter `minlevel` is an integer. `minlevel=1` means that all clusters in the tree are given out by `hclusttreeCBI` or `disthclusttreeCBI`, including one-point clusters (but excluding the cluster with all points). `minlevel=2` excludes the one-point clusters. `minlevel=3` excludes the two-point cluster which has been merged first, and increasing the value of `minlevel` by 1 in all further steps means that the remaining earliest formed cluster is excluded.

Parameter `G` is a vector of integers. Number of clusters or numbers of clusters used by `mclustBIC`. If `G` has more than one entry, the number of clusters is estimated by the BIC.

Parameter `modelNames` is a vector of string. Models for covariance matrices, see documentation of `mclustBIC`.

Parameter `nnk` is an integer. Tuning constant for `NNclean`, which is used to estimate the initial noise for `noisemclustCBI` and `distnoisemclustCBI`. See parameter `k` in the documentation of `NNclean`. `nnk=0` means that no noise component is fitted.

Parameter `hcmodel` is a string or `NULL`. Determines the initialization of the EM-algorithm for `mclustBIC`. Documented in `hc`.

Parameter `Vinv` is numeric. See documentation of `mclustBIC`.

Parameter `summary.out` is logical. If `TRUE`, the result of `summary.mclustBIC` is added as component `mclustsummary` to the output of `noisemclustCBI` and `distnoisemclustCBI`.

Parameter `mds.method` is "classical", "kruskal" or "sammon". Determines the multidimensional scaling method to compute Euclidean data from a dissimilarity matrix. See `cmdscale`, `isoMDS` and `sammon`.
mdsdim integer. Dimensionality of MDS solution.
points.out logical. If TRUE, the matrix of MDS points is added as component points to the output of noisemclustCBI.
usepam logical. If TRUE, the function pam is used for clustering, otherwise clara, pam is better, clara is faster.
diss logical. If TRUE, data will be considered as a dissimilarity matrix. In claraCBI, this requires usepam=TRUE.
krange vector of integers. Numbers of clusters to be compared.
trim numeric between 0 and 1. Proportion of data points trimmed, i.e., assigned to noise. See tclust in the tclust package, trimkmeans.
eps numeric. The radius of the neighborhoods to be considered by dbscan.
MinPts integer. How many points have to be in a neighborhood so that a point is considered to be a cluster seed? See documentation of dbscan.
clustercut numeric between 0 and 1. If fixmahal is used for fuzzy clustering, a crisp partition is generated and points with cluster membership values above clustercut are considered as members of the corresponding cluster.
mergemethod method for merging Gaussians, passed on as method to mergenormals.
cutoff numeric between 0 and 1, tuning constant for mergenormals.
... further parameters to be transferred to the original clustering functions (not required).

Details

All these functions call clustering methods implemented in R to cluster data and to provide output in the format required by clusterboot. Here is a brief overview. For further details see the help pages of the involved clustering methods.

kmeansCBI an interface to the function kmeansruns calling kmeans for k-means clustering. (kmeansruns allows the specification of several random initializations of the k-means algorithm and estimation of k by the Calinski-Harabasz index or the average silhouette width.)
hclustCBI an interface to the function hclust for agglomerative hierarchical clustering with noise component (see parameter noisecut above). This function produces a partition and assumes a cases*variables matrix as input.
hclusttreeCBI an interface to the function hclust for agglomerative hierarchical clustering. This function gives out all clusters belonging to the hierarchy (upward from a certain level, see parameter minlevel above).
disthclustCBI an interface to the function hclust for agglomerative hierarchical clustering with noise component (see parameter noisecut above). This function produces a partition and assumes a dissimilarity matrix as input.
noisemclustCBI an interface to the function mclustBIC, for normal mixture model based clustering. Warning: mclustBIC often has problems with multiple points. In clusterboot, it is recommended to use this together with multipleboot=FALSE.
distnoisemclustCBI an interface to the function mclustBIC for normal mixture model based clustering. This assumes a dissimilarity matrix as input and generates a data matrix by multidimensional scaling first. Warning: mclustBIC often has problems with multiple points. In clusterboot, it is recommended to use this together with multipleboot=FALSE.
### Value

All interface functions return a list with the following components (there may be some more, see `summary.out` and `points.out` above):

- **result**: clustering result, usually a list with the full output of the clustering method (the precise format doesn’t matter); whatever you want to use later.
- **nc**: number of clusters. If some points don’t belong to any cluster, these are declared "noise". `nc` includes the "noise cluster", and there should be another component `nccl`, being the number of clusters not including the noise cluster.
- **clusterlist**: this is a list consisting of a logical vectors of length of the number of data points (`n`) for each cluster, indicating whether a point is a member of this cluster (`TRUE`) or not. If a noise cluster is included, it should always be the last vector in this list.
- **partition**: an integer vector of length `n`, partitioning the data. If the method produces a partition, it should be the clustering. This component is only used for plots, so you could do something like `rep(1,n)` for non-partitioning methods. If a noise cluster is included, `nc=nccl+1` and the noise cluster is cluster no. `nc`.
- **clustermethod**: a string indicating the clustering method.

The output of some of the functions has further components:

- **nccl**: see `nc` above.
kmeansruns

nnk by noisemclustCBI and distnoisemclustCBI, see above.
initnoise logical vector, indicating initially estimated noise by NNClean, called by noisemclustCBI and distnoisemclustCBI.
noise logical. TRUE if points were classified as noise/outliers by disthclustCBI.

Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

See Also
clusterboot, dist, kmeans, kmeansruns, hclust, mclustBIC, pam, pamk, clara, trimkmeans, dbscan, fixmahal, tclust, pdfCluster

Examples

options(digits=3)
set.seed(20000)
face <- rFace(50,dMnoN2,dNoEyN0,p=2)
dbs <- dbscanCBI(face,eps=1.5,MinPts=4)
dhc <- disthclustCBI(dist(face),method="average",k=1.5,noisecut=2)
table(dbs$partition,dhc$partition)
dm <- mergenormCBI(face,G=10,modelNames="EEE",nnk=2)
dtc <- tclustCBI(face,6,trim=0.1,restr.fact=500)
table(dm$partition,dtc$partition)

kmeansruns k-means with estimating k and initialisations

Description
This calls the function kmeans to perform a k-means clustering, but initializes the k-means algorithm several times with random points from the data set as means. Furthermore, it is more robust against the occurrence of empty clusters in the algorithm and it estimates the number of clusters by either the Calinski Harabasz index (calinhara) or average silhouette width (see pam.object). The Duda-Hart test (dudahart2) is applied to decide whether there should be more than one cluster (unless 1 is excluded as number of clusters).

Usage

kmeansruns(data,krange=2:10,criterion="ch",
iter.max=100,runs=100,
scaledata=FALSE,alpha=0.001,
critout=FALSE,plot=FALSE,...)
**Arguments**

- **data**: 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).
- **krange**: integer vector. Numbers of clusters which are to be compared by the average silhouette width criterion. Note: average silhouette width and Calinski-Harabasz can’t estimate number of clusters nc=1. If 1 is included, a Duda-Hart test is applied and 1 is estimated if this is not significant.
- **criterion**: one of "asw" or "ch". Determines whether average silhouette width or Calinski-Harabasz is applied.
- **iter.max**: integer. The maximum number of iterations allowed.
- **runs**: integer. Number of starts of the k-means algorithm.
- **scaledata**: logical. If TRUE, the variables are centered and scaled to unit variance before execution.
- **alpha**: numeric between 0 and 1, tuning constant for dudahart2 (only used for 1-cluster test).
- **critout**: logical. If TRUE, the criterion value is printed out for every number of clusters.
- **plot**: logical. If TRUE, every clustering resulting from a run of the algorithm is plotted.
- **...**: further arguments to be passed on to kmeans.

**Value**

The output of the optimal run of the kmeans-function with added components bestk and crit. A list with components

- **cluster**: A vector of integers indicating the cluster to which each point is allocated.
- **centers**: A matrix of cluster centers.
- **withinss**: The within-cluster sum of squares for each cluster.
- **size**: The number of points in each cluster.
- **bestk**: The optimal number of clusters.
- **crit**: Vector with values of the criterion for all used numbers of clusters (0 if number not tried).

**Author(s)**

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

**References**


See Also

kmeans, pamk, calinhara, dudahart2)

Examples

options(digits=3)
set.seed(20000)

face <- rFace(50, dMoNo=2, dNoEy=0, p=2)

pk <- kmeansruns(face, krange=1:5, critout=TRUE, runs=2, criterion="asw")
pkc <- kmeansruns(face, krange=1:5, critout=TRUE, runs=2, criterion="ch")

Usage

lcmixed( formula = .~. , continuous, discrete, ppdim,
               diagonal = TRUE, pred.ordinal=FALSE, printlik=FALSE )

Arguments

formula a formula to specify response and explanatory variables. For lcmixed this always has the form x~1, where x is a matrix or data frame of all variables to be involved, because regression and explanatory variables are not implemented.

continuous number of continuous variables. Note that the continuous variables always need to be the first variables in the matrix or data frame.

discrete number of categorical variables. Always the last variables in the matrix or data frame. Note that categorical variables always must be coded as integers 1,2,3, etc. without interruption.

ppdim vector of integers specifying the number of (in the data) existing categories for each categorical variable.
diagonal logical. If TRUE, Gaussian models are fitted restricted to diagonal covariance matrices. Otherwise, covariance matrices are unrestricted. TRUE is consistent with the "within class independence" assumption for the multinomial variables.

pred.ordinal logical. If FALSE, the within-component predicted value for categorical variables is the probability mode, otherwise it is the mean of the standard (1,2,3,...) scores, which may be better for ordinal variables.

printlik logical. If TRUE, the loglikelihood is printed out whenever computed.

Details

The data need to be organised case-wise, i.e., if there are categorical variables only, and 15 cases with values c(1,1,2) on the 3 variables, the data matrix needs 15 rows with values 1 1 2.

General documentation on flexmix methods can be found in Chapter 4 of Friedrich Leisch's "FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R", https://CRAN.R-project.org/package=flexmix

Value

An object of class FLXMC (not documented; only used internally by flexmix).

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche

References


See Also

flexmixedruns, flexmix, flexmix-class, discrete.recode, which recodes a dataset into the format required by lcmixed

Examples

```r
set.seed(12233)
options(digits=3)
require(MASS)
require(flexmix)
data(Cars93)
Cars934 <- Cars93[,c(3,5,8,10)]
cc <-
discrete.recode(Cars934,xvsorted=FALSE,continuous=c(2,3),discrete=c(1,4))
fcc <- flexmix(cc$data=1,k=2,
model=lcmixed(continuous=2,discrete=2,ppdim=c(6,3),diagonal=TRUE))
summary(fcc)
```
localshape

Local shape matrix

Description

This computes a matrix formalising 'local shape', i.e., aggregated standardised variance/covariance in a Mahalanobis neighbourhood of the data points. This can be used for finding clusters when used as one of the covariance matrices in Invariant Coordinate Selection (function ics in package ICS), see Hennig’s discussion and rejoinder of Tyler et al. (2009).

Usage

localshape(xdata, proportion=0.1, mscatter="mcd", mcdalpha=0.8, covstandard="det")

Arguments

- **xdata**: objects times variables data matrix.
- **proportion**: proportion of points to be considered as neighbourhood.
- **mscatter**: "mcd" or "cov"; specified minimum covariance determinant or classical covariance matrix to be used for Mahalanobis distance computation.
- **mcdalpha**: if mscatter="mcd", this is the alpha parameter to be used by the MCD covariance matrix, i.e. one minus the asymptotic breakdown point, see covMcd.
- **covstandard**: one of "trace", "det" or "none", determining by what constant the pointwise neighbourhood covariance matrices are standardised. "det" makes the affine equivariant, as noted in the discussion rejoinder of Tyler et al. (2009).

Value

The local shape matrix.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche](http://www.homepages.ucl.ac.uk/~ucakche)

References


Examples

```
options(digits=3)
data(iris)
localshape(iris[,5], mscatter="cov")
```
**mahalanodisc**  

*Mahalanobis for AWC*

---

**Description**

Vector of Mahalanobis distances or their root. For use in awcoord only.

**Usage**

```r
mahalanodisc(x, mg, covg, modus="square")
```

**Arguments**

- `x`: numerical data matrix.
- `mg`: mean vector.
- `covg`: covariance matrix.
- `modus`: "md" (roots of Mahalanobis distances) or "square" (original squared form of Mahalanobis distances).

**Details**

The covariance matrix is inverted by use of `solvecov`, which can be expected to give reasonable results for singular within-class covariance matrices.

**Value**

vector of (rooted) Mahalanobis distances.

**Author(s)**

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

**See Also**

`awcoord`, `solvecov`

**Examples**

```r
options(digits=3)
x <- cbind(rnorm(50), rnorm(50))
mahalanodisc(x, c(0, 0), cov(x))
mahalanodisc(x, c(0, 0), matrix(0, ncol=2, nrow=2))
```
mahalanofix  

Mahalanobis distances from center of indexed points

Description

Computes the vector of (classical or robust) Mahalanobis distances of all points of \( x \) to the center of the points indexed (or weighted) by \( gv \). The latter also determine the covariance matrix.

Thought for use within fixmahal.

Usage

```r
mahalanofix(x, n = nrow(as.matrix(x)), p = ncol(as.matrix(x)), gv = rep(1, times = n), cmax = 1e+10, method = "ml")
```

```r
mahalanofuz(x, n = nrow(as.matrix(x)), p = ncol(as.matrix(x)),
             gv = rep(1, times=n), cmax = 1e+10)
```

Arguments

- **\( x \)**: a numerical data matrix, rows are points, columns are variables.
- **\( n \)**: positive integer. Number of points.
- **\( p \)**: positive integer. Number of variables.
- **\( gv \)**: for mahalanofix a logical or 0-1 vector of length \( n \). For mahalanofuz a numerical vector with values between 0 and 1.
- **cmax**: positive number. used in solvecov if covariance matrix is singular.
- **method**: "ml", "classical", "mcd" or "mve". Method to compute the covariance matrix estimator. See cov.rob, fixmahal.

Details

solvecov is used to invert the covariance matrix. The methods "mcd" and "mve" in mahalanofix do not work properly with point constellations with singular covariance matrices!

Value

A list of the following components:

- **md**: vector of Mahalanobis distances.
- **mg**: mean of the points indexed by \( gv \), weighted mean in mahalanofuz.
- **covg**: covariance matrix of the points indexed by \( gv \), weighted covariance matrix in mahalanofuz.
- **covinv**: covg inverted by solvecov.
- **coll**: logical. If TRUE, covg has been (numerically) singular.
**Note**
Methods "mcd" and "mve" require library lqs.

**Author(s)**
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

**See Also**
fixmahal, solvecov, cov.rob

**Examples**
```
x <- c(1,2,3,4,5,6,7,8,9,10)
y <- c(1,2,3,8,7,6,5,8,9,10)
mahalanofix(cbind(x,y),gv=c(0,0,0,1,1,1,1,0,0))
mahalanofix(cbind(x,y),gv=c(0,0,0,1,1,1,1,0,0))
mahalanofix(cbind(x,y),gv=c(0,0,0,1,1,1,1,0,0),method="mcd")
mahalanofuz(cbind(x,y),gv=c(0,0,0,5,0.5,5,1,1,0.5,0.5,0))
```

**Description**
Generates an initial configuration of startn points from dataset x for the fixmahal fixed point iteration. Thought only for use within fixmahal.

**Usage**
mahalconf(x, no, startn, covall, plot)

**Arguments**
- **x** numerical matrix. Rows are points, columns are variables.
- **no** integer between 1 and nrow(x). Number of the first point of the configuration.
- **startn** integer between 1 and nrow(x).
- **covall** covariance matrix for the computation of the first Mahalanobis distances.
- **plot** a string. If equal to "start" or "both", the first two variables and the first ncol(x)+1 points are plotted.

**Details**
mahalconf first chooses the p (number of variables) nearest points to point no. no in terms of the Mahalanobis distance w.r.t. covall, so that there are p + 1 points. In every further step, the covariance matrix of the current configuration is computed and the nearest point in terms of the new Mahalanobis distance is added. solvecov is used to invert singular covariance matrices.
Value

A logical vector of length nrow(x).

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

See Also

fixmahal, solvecov

Examples

set.seed(4634)
face <- rFace(600, dMNo=2, dNoEY=0, p=2)
mahalconf(face, no=200, startn=20, covall=cov(face), plot="start")

mergenormals

Clustering by merging Gaussian mixture components

Description

Clustering by merging Gaussian mixture components; computes all methods introduced in Hennig (2010) from an initial mclust clustering. See details section for details.

Usage

mergenormals(xdata = NULL, mclustsummary = NULL,
  clustering = NULL, probs = NULL, muarray = NULL,
  Sigmaarray = NULL, z = NULL,
  method = NULL, cutoff = NULL, by = 0.005,
  numberstop = NULL, renumber = TRUE, M = 50, ...)

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

## S3 method for class 'summary.mergenorm'
print(x, ...)

Arguments

xdata
data (something that can be coerced into a matrix).

mclustsummary
output object from summary.mclustBIC for xdata. Either mclustsummary or all of clustering, probs, muarray, Sigmaarray and z need to be specified (the latter are obtained from mclustsummary if they are not provided). I am not aware of restrictions of the usage of mclustBIC to produce an initial clustering; covariance matrix models can be restricted and a noise component can be included if desired, although I have probably not tested all possibilities.
clustering  vector of integers. Initial assignment of data to mixture components.
probs    vector of component proportions (for all components; should sum up to one).
muarray  matrix of component means (rows).
Sigmaarray array of component covariance matrices (third dimension refers to component number).
z        matrix of observation- (row-)wise posterior probabilities of belonging to the components (columns).
method   one of "bhat", "ridge.uni", "ridge.ratio", "demp", "dipuni", "diptantrum", "predictive". See details.
cutoff   numeric between 0 and 1. Tuning constant, see details and Hennig (2010). If not specified, the default values given in (9) in Hennig (2010) are used.
by       real between 0 and 1. Interval width for density computation along the ridgeline, used for methods "ridge.uni" and "ridge.ratio". Methods "dipuni" and "diptantrum" require ridgeline computations and use it as well.
numberstop integer. If specified, cutoff is ignored and components are merged until the number of clusters specified here is reached.
renumber  logical. If TRUE merged clusters are renumbered from 1 to their number. If not, numbers of the original clustering are used (numbers of components that were merged into others then will not appear).
M        integer. Number of times the dataset is divided into two halves. Used if method="predictive".
...      additional optional parameters to pass on to ridgeline.diagnosis or mixpredictive (in mergenormals).
object   object of class mergenorm, output of mergenormals.
x        object of class summary.mergenorm, output of summary.mergenorm.

Details
Mixture components are merged in a hierarchical fashion. The merging criterion is computed for all pairs of current clusters and the two clusters with the highest criterion value (lowest, respectively, for method="predictive") are merged. Then criterion values are recomputed for the merged cluster. Merging is continued until the criterion value to merge is below (or above, for method="predictive") the cutoff value. Details are given in Hennig (2010). The following criteria are offered, specified by the method-argument.

"ridge.uni" components are only merged if their mixture is unimodal according to Ray and Lindsay’s (2005) ridgeline theory, see ridgeline.diagnosis. This ignores argument cutoff.

"ridge.ratio" ratio between density minimum between components and minimum of density maxima according to Ray and Lindsay’s (2005) ridgeline theory, see ridgeline.diagnosis.

"bhat" Bhattacharyya upper bound on misclassification probability between two components, see bhattacharyya.matrix.

"demp" direct estimation of misclassification probability between components, see Hennig (2010).

"dipuni" this uses method="ridge.ratio" to decide which clusters to merge but stops merging according to the p-value of the dip test computed as in Hartigan and Hartigan (1985), see dip.test.
"dipatantrum" as "dipuni", but p-value of dip test computed as in Tantrum, Murua and Stuetzle (2003), see dip.p.tantrum.

"predictive" this uses method="demp" to decide which clusters to merge but stops merging according to the value of prediction strength (Tibshirani and Walther, 2005) as computed in mixpredictive.

Value

mergenormals gives out an object of class mergenorm, which is a List with components

- clustering integer vector. Final clustering.
- clusternumbers vector of numbers of remaining clusters. These are given in terms of the original clusters even of renumber=TRUE, in which case they may be needed to understand the numbering of some further components, see below.
- defunct.components vector of numbers of components that were "merged away".
- valuemerged vector of values of the merging criterion (see details) at which components were merged.
- mergedtonumbers vector of numbers of clusters to which the original components were merged.
- parameters a list, if mclust.summary was provided. Entry no. i refers to number i in clusternumbers. The list entry i contains the parameters of the original mixture components that make up cluster i, as extracted by extract.mixturepars.
- predvalues vector of prediction strength values for clusternumbers from 1 to the number of components in the original mixture, if method="predictive". See mixpredictive.
- orig.decisionmatrix square matrix with entries giving the original values of the merging criterion (see details) for every pair of original mixture components.
- new.decisionmatrix square matrix as orig.decisionmatrix, but with final entries; numbering of rows and columns corresponds to clusternumbers; all entries corresponding to other rows and columns can be ignored.
- probs final cluster values of probs (see arguments) for merged components, generated by (potentially repeated) execution of mergeparameters out of the original ones. Numbered according to clusternumbers.
- muarray final cluster means, analogous to probs.
- Sigmaarray final cluster covariance matrices, analogous to probs.
- z final matrix of posterior probabilities of observations belonging to the clusters, analogous to probs.
- noise logical. If TRUE, there was a noise component fitted in the initial mclust clustering (see help for initialization in mclustBIC). In this case, a cluster number 0 indicates noise. noise is ignored by the merging methods and kept as it was originally.
- method as above.
- cutoff as above.
summary.mergenorm gives out a list with components clustering, clusternumbers, defunct.components, valuemerged as above, plus onc (original number of components) and mnc (number of clusters after merging).

Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


Examples
```r
require(mclust)
require(MASS)
options(digits=3)
data(crabs)
dc <- crabs[,4:8]
cm <- mclustBIC(crabs[,4:8],G=9,modelNames="EEE")
scm <- summary(cm,crabs[,4:8])
cmnbhat <- mergenormals(crabs[,4:8],scm,method="bhat")
summary(cmnbhat)
cmndemp <- mergenormals(crabs[,4:8],scm,method="demp")
summary(cmndemp)
# Other methods take a bit longer, but try them!
# The values of by and M below are still chosen for reasonably fast execution.
# cmnrr <- mergenormals(crabs[,4:8],scm,method="ridge.ratio",by=0.05)
# cdm <- mergenormals(crabs[,4:8],scm,method="dip.tantrum",by=0.05)
# cmp <- mergenormals(crabs[,4:8],scm,method="predictive",M=3)
```

mergeparameters

*New parameters from merging two Gaussian mixture components*

Description
Re-computes pointwise posterior probabilities, mean and covariance matrix for a mixture component obtained by merging two mixture components in a Gaussian mixture.
Usage

mergeparameters(xdata, j1, j2, probs, muarray, Sigmaarray, z)

Arguments

- **xdata**: data (something that can be coerced into a matrix).
- **j1**: integer. Number of first mixture component to be merged.
- **j2**: integer. Number of second mixture component to be merged.
- **probs**: vector of component proportions (for all components; should sum up to one).
- **muarray**: matrix of component means (rows).
- **Sigmaarray**: array of component covariance matrices (third dimension refers to component number).
- **z**: matrix of observation- (row-)wise posterior probabilities of belonging to the components (columns).

Value

List with components

- **probs**: see above; sum of probabilities for original components j1 and j2 is now `probs[j1]`. Note that generally, also for the further components, values for the merged component are in place j1 and values in place j2 are not changed. This means that in order to have only the information for the new mixture after merging, the entries in places j2 need to be suppressed.
- **muarray**: see above; weighted mean of means of component j1 and j2 is now in place j1.
- **Sigmaarray**: see above; weighted covariance matrix handled as above.
- **z**: see above; original entries for columns j1 and j2 are summed up and now in column j1.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

References


Examples

```r
options(digits=3)
set.seed(98765)
require(mclust)
iriss <- iris[sample(150,20),-5]
irisBIC <- mclustBIC(iriss)
siris <- summary(irisBIC,iriss)
probs <- siris$parameters$pro
```
**minsize**

```
muarray <- siris$parameters$mean
Sigmaarray <- siris$parameters$variance$sigma
z <- siris$z
mpi <- mergeparameters(iriss, 1, 2, probs, muarray, Sigmaarray, z)
mpi$probs
mpi$muarray
```

---

**minsize**

Minimum size of regression fixed point cluster

**Description**

Computes the minimum size of a fixed point cluster (FPC) which is found at least `mtf` times with approximated probability `prob` by `ir` fixed point iterations of `fixreg`.

Thought for use within `fixreg`.

**Usage**

`minsize(n, p, ir, mtf, prob = 0.5)`

**Arguments**

- `n` positive integer. Total number of points.
- `p` positive integer. Number of independent variables.
- `ir` positive integer. Number of fixed point iterations.
- `mtf` positive integer.
- `prob` numerical between 0 and 1.

**Details**

The computation is based on the binomial distribution with probability given by `clusexpect` with `ir=1`.

**Value**

An integer.

**Author(s)**

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

**References**

See Also

fixreg, clusexpect, itnumber

Examples

minsize(500,4,7000,2)

mixdens

Density of multivariate Gaussian mixture, mclust parameterisation

Description

Computes density values for data from a mixture of multivariate Gaussian distributions with parameters based on the way models are specified and parameters are stored in package mclust.

Usage

mixdens(modelName, data, parameters)

Arguments

modelName

an mclust model name. See mclustModelNames.

data

data matrix; density values are computed for every observation (row).

parameters

parameters of Gaussian mixture in the format used in the output of summary.mclustBIC.

Value

Vector of density values for the observations.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

Examples

set.seed(98765)
require(mclust)
iriss <- iris[sample(150,20),-5]
irisBIC <- mclustBIC(iriss)
siris <- summary(irisBIC,iriss)
siris <- summary(irisBIC,iriss)
round(mixdens(siris$modelName,iriss,siris$parameters),digits=2)
mixpredictive  

Prediction strength of merged Gaussian mixture

Description

Computes the prediction strength of clustering by merging Gaussian mixture components, see `mergenormals`. The predictive strength is defined according to Tibshirani and Walther (2005), carried out as described in Hennig (2010), see details.

Usage

```r
mixpredictive(xdata, Gcomp, Gmix, M=50, ...)
```

Arguments

- `xdata`: data (something that can be coerced into a matrix).
- `Gcomp`: integer. Number of components of the underlying Gaussian mixture.
- `Gmix`: integer. Number of clusters after merging Gaussian components.
- `M`: integer. Number of times the dataset is divided into two halves.
- `...`: further arguments that can potentially arrive in calls but are currently not used.

Details

The prediction strength for a certain number of clusters `Gmix` under a random partition of the dataset in halves A and B is defined as follows. Both halves are clustered with `Gmix` clusters. Then the points of A are classified to the clusters of B. This is done by use of the maximum a posteriori rule for mixtures as in Hennig (2010), differently from Tibshirani and Walther (2005). A pair of points A in the same A-cluster is defined to be correctly predicted if both points are classified into the same cluster on B. The same is done with the points of B relative to the clustering on A. The prediction strength for each of the clusterings is the minimum (taken over all clusters) relative frequency of correctly predicted pairs of points of that cluster. The final mean prediction strength statistic is the mean over all 2M clusterings.

Value

List with components

- `predcorr`: vector of length M with relative frequencies of correct predictions (clusterwise minimum).
- `mean.pred`: mean of `predcorr`.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)
References


See Also

`prediction.strength` for Tibshirani and Walther's original method. `mergenormals` for the clustering method applied here.

Examples

```r
set.seed(98765)
iris <- iris[sample(150,20),-5]
mp <- mixpredictive(iris,2,2,M=2)
```

**mvdcoord**

*Mean/variance differences discriminant coordinates*

Description

Discriminant projections as defined in Young, Marco and Odell (1987). The principle is to maximize the projection of a matrix consisting of the differences between the means of all classes and the first mean and the differences between the covariance matrices of all classes and the first covariance matrix.

Usage

```r
mvdcoord(xd, clvecd, clnum=1, sphere="mcd", ...)
```

Arguments

- `xd` the data matrix; a numerical object which can be coerced to a matrix.
- `clvecd` integer vector of class numbers; length must equal `nrow(xd)`.
- `clnum` integer. Number of the class to which all differences are computed.
- `sphere` a covariance matrix or one of "mve", "mcd", "classical", "none". The matrix used for sphering the data. "mcd" and "mve" are robust covariance matrices as implemented in `cov.rob`. "classical" refers to the classical covariance matrix. "none" means no sphering and use of the raw data.
- `...` no effect
Value

List with the following components

ev eigenvalues in descending order.
units columns are coordinates of projection basis vectors. New points x can be projected onto the projection basis vectors by x %*% units
proj projections of xd onto units.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


See Also

plotcluster for straightforward discriminant plots. discrproj for alternatives. rFace for generation of the example data used below.

Examples

```r
set.seed(4634)
face <- rFace(300,dMoN=2,dNoE=0,p=3)
grface <- as.integer(attr(face,"grouping"))
mcf <- mvdcoord(face,grface)
plot(mcf$proj,col=grface)
# ...done in one step by function plotcluster.
```

ncoord

*Neighborhood based discriminant coordinates*

Description

Neighborhood based discriminant coordinates as defined in Hastie and Tibshirani (1996) and a robustified version as defined in Hennig (2003). The principle is to maximize the projection of a between classes covariance matrix, which is defined by averaging the between classes covariance matrices in the neighborhoods of all points.

Usage

```r
ncoord(xd, clvecd, nn=50, weighted=FALSE,
       sphere="mcd", orderall=TRUE, countmode=1000, ...)
```
Arguments

xd
the data matrix; a numerical object which can be coerced to a matrix.

clvecd
integer vector of class numbers; length must equal nrow(xd).

nn
integer. Number of points which belong to the neighborhood of each point (including the point itself).

weighted
logical. FALSE corresponds to the original method of Hastie and Tibshirani (1996). If TRUE, the between classes covariance matrices B are weighted by w/trace B, where w is some weight depending on the sizes of the classes in the neighborhood. Division by trace B reduces the effect of outliers. TRUE corresponds to WNC as defined in Hennig (2003).

sphere
a covariance matrix or one of "mve", "mcd", "classical", "none". The matrix used for sphering the data. "mcd" and "mve" are robust covariance matrices as implemented in cov.rob. "classical" refers to the classical covariance matrix. "none" means no sphering and use of the raw data.

orderall
logical. By default, the neighborhoods are computed by ordering all points each time. If FALSE, the neighborhoods are computed by selecting nn times the nearest point from the remaining points, which may be faster sometimes.

countmode
optional positive integer. Every countmode algorithm runs ncoord shows a message.

Value

List with the following components

ev
eigenvalues in descending order.

units
columns are coordinates of projection basis vectors. New points x can be projected onto the projection basis vectors by x * units

proj
projections of xd onto units.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


See Also

plotcluster for straight forward discriminant plots. discrproj for alternatives. rFace for generation of the example data used below.
neginc

Examples

set.seed(4634)
face <- rFace(600, dMNo=2, dNoEy=0)
grface <- as.integer(attr(face, "grouping"))
ncf <- ncoord(face, grface)
plot(ncf$proj, col=grface)
ncf2 <- ncoord(face, grface, weighted=TRUE)
plot(ncf2$proj, col=grface)
# ... done in one step by function plotcluster.

Description

Cluster validity index based on the neg-entropy distances of within-cluster distributions to normal distribution, see Lago-Fernandez and Corbacho (2010).

Usage

neginc(x, clustering)

Arguments

x something that can be coerced into a numerical matrix. Euclidean dataset.
clustering vector of integers with length = nrow(x); indicating the cluster for each observation.

Value

Index value, see Lago-Fernandez and Corbacho (2010). The lower (i.e., the more negative) the better.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


Examples

options(digits=3)
iriss <- as.matrix(iris[c(1:10, 51:55, 101:105), -5])
irisc <- as.numeric(iris[c(1:10, 51:55, 101:105), 5])
neginc(iriss, irisc)
nselectboot  

Selection of the number of clusters via bootstrap

Description

Selection of the number of clusters via bootstrap as explained in Fang and Wang (2012). Several times 2 bootstrap samples are drawn from the data and the number of clusters is chosen by optimising an instability estimation from these pairs.

In principle all clustering methods can be used that have a CBI-wrapper, see clusterboot, kmeansCBI. However, the currently implemented classification methods are not necessarily suitable for all of them, see argument classification.

Usage

nselectboot(data, B=50, distances=inherits(data, "dist"),
            clustermethod=NULL,
            classification="averagedist", krange=2:10,
            count=FALSE, nnk=1, ...)

Arguments

data  something that can be coerced into a matrix. The data matrix - either an n*p-data matrix (or data frame) or an n*n-dissimilarity matrix (or dist-object).

B     integer. Number of resampling runs.

distances  logical. If TRUE, the data is interpreted as dissimilarity matrix. If data is a dist-object, distances=TRUE automatically, otherwise distances=FALSE by default. This means that you have to set it to TRUE manually if data is a dissimilarity matrix.

clustermethod an interface function (the function name, not a string containing the name, has to be provided!). This defines the clustering method. See the "Details"-section of clusterboot and kmeansCBI for the format. Clustering methods for nselectboot must have a k-argument for the number of clusters and must otherwise follow the specifications in clusterboot.

classification  string. This determines how non-clustered points are classified to given clusters. Options are explained in classifdist (if distances=TRUE) and classifnp (otherwise). Certain classification methods are connected to certain clustering methods. classification="averagedist" is recommended for average linkage, classification="centroid" is recommended for k-means, clara and pam, classification="knn" with nnk=1 is recommended for single linkage and classification="qda" is recommended for Gaussian mixtures with flexible covariance matrices.

krange    integer vector; numbers of clusters to be tried.

count     logical. If TRUE, numbers of clusters and bootstrap runs are printed.

nnk       number of nearest neighbours if classification="knn", see classifdist (if distances=TRUE) and classifnp (otherwise).

... arguments to be passed on to the clustering method.
Value

nselectboot returns a list with components kopt, stabk, stab.

- **kopt**: optimal number of clusters.
- **stabk**: mean instability values for numbers of clusters.
- **stab**: matrix of instability values for all bootstrap runs and numbers of clusters.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

References


See Also

classifdist, classifnp, clusterboot, kmeansCBI

Examples

```r
set.seed(20000)
face <- rFace(50, dMoNo=2, dNoEy=0, p=2)
nselectboot(dist(face), B=2, clustermethod=distclustCBI,
  method="average", krange=5:7)
nselectboot(dist(face), B=2, clustermethod=claraCBI,
  classification="centroid", krange=5:7)
nselectboot(face, B=2, clustermethod=kmeansCBI,
  classification="centroid", krange=5:7)
# Of course use larger B in a real application.
```

Description

This calls the function pam or clara to perform a partitioning around medoids clustering with the number of clusters estimated by optimum average silhouette width (see pam.object) or Calinski-Harabasz index (calinhara). The Duda-Hart test (dudahart2) is applied to decide whether there should be more than one cluster (unless 1 is excluded as number of clusters or data are dissimilarities).

Usage

```r
pamk(data, krange=2:10, criterion="asw", usepam=TRUE,
  scaling=FALSE, alpha=0.001, diss=inherits(data, "dist"),
  critout=FALSE, ns=10, seed=NULL, ...)
```
Arguments

data
- a data matrix or data frame or something that can be coerced into a matrix, or dissimilarity matrix or object. See **pam** for more information.

krange
- integer vector. Numbers of clusters which are to be compared by the average silhouette width criterion. Note: average silhouette width and Calinski-Harabasz can't estimate number of clusters nc=1. If 1 is included, a Duda-Hart test is applied and 1 is estimated if this is not significant.

criterion
- one of "asw", "multiasw" or "ch". Determines whether average silhouette width (as given out by **pam**/**clara**, or as computed by **distcritmulti** if "multiasw" is specified; recommended for large data sets with usepam=FALSE) or Calinski-Harabasz is applied. Note that the original Calinski-Harabasz index is not defined for dissimilarities; if dissimilarity data is run with criterion="ch", the dissimilarity-based generalisation in Hennig and Liao (2013) is used.

usepam
- logical. If TRUE, **pam** is used, otherwise **clara** (recommended for large datasets with 2,000 or more observations; dissimilarity matrices can not be used with **clara**).

scaling
- either a logical value or a numeric vector of length equal to the number of variables. If scaling is a numeric vector with length equal to the number of variables, then each variable is divided by the corresponding value from scaling. If scaling is TRUE then scaling is done by dividing the (centered) variables by their root-mean-square, and if scaling is FALSE, no scaling is done.

alpha
- numeric between 0 and 1, tuning constant for **dudahart2** (only used for 1-cluster test).

diss
- logical flag: if TRUE (default for dist or dissimilarity-objects), then data will be considered as a dissimilarity matrix (and the potential number of clusters 1 will be ignored). If FALSE, then data will be considered as a matrix of observations by variables.

critout
- logical. If TRUE, the criterion value is printed out for every number of clusters.

ns
- passed on to **distcritmulti** if criterion="multiasw".

seed
- passed on to **distcritmulti** if criterion="multiasw".

... further arguments to be transferred to **pam** or **clara**.

Value

A list with components

- **pamobject** The output of the optimal run of the **pam**-function.
- **nc** the optimal number of clusters.
- **crit** vector of criterion values for numbers of clusters. crit[1] is the p-value of the Duda-Hart test if 1 is in krange and diss=FALSE.

Note

**clara** and **pam** can handle NA-entries (see their documentation) but **dudahart2** cannot. Therefore NA should not occur if 1 is in krange.
piridge

Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References

See Also
pam, clara, distcritmulti

Examples
options(digits=3)
set.seed(20000)
f <- rFace(50,dMoNo=2,dNoEy=0,p=2)
ik1 <- pamk(f,krange=1:5,criterion="asw",critout=TRUE)
ik2 <- pamk(f,krange=1:5,criterion="multiasw",ns=2,critout=TRUE)
# "multiasw" is better for larger data sets, use larger ns then.
ilk3 <- pamk(f,krange=1:5,criterion="ch",critout=TRUE)

piridge

Ridgeline Pi-function

Description
The Pi-function is given in (6) in Ray and Lindsay, 2005. Equating it to the mixture proportion yields locations of two-component Gaussian mixture density extrema.

Usage
piridge(alpha, mu1, mu2, Sigma1, Sigma2, showplot=FALSE)

Arguments
alpha sequence of values between 0 and 1 for which the Pi-function is computed.
mu1 mean vector of component 1.
u2 mean vector of component 2.
Sigma1 covariance matrix of component 1.
Sigma2 covariance matrix of component 2.
showplot logical. If TRUE, the Pi-function is plotted against alpha.
Value
Vector of values of the Pi-function for values of alpha.

Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References

Examples
q <- piridge(seq(0,1,0.1),c(1,1),c(2,5),diag(2),diag(2))

pireidge.zeros  Extrema of two-component Gaussian mixture

Description
By use of the Pi-function in Ray and Lindsay, 2005, locations of two-component Gaussian mixture density extrema or saddlepoints are computed.

Usage
piridge.zeroses(prop, mu1, mu2, Sigma1, Sigma2, alphamin=0, alphamax=1, by=0.001)

Arguments
prop proportion of mixture component 1.
mu1 mean vector of component 1.
mu2 mean vector of component 2.
Sigma1 covariance matrix of component 1.
Sigma2 covariance matrix of component 2.
alphamin minimum alpha value.
alphamax maximum alpha value.
by interval between alpha-values where to look for extrema.

Value
list with components
number.zeroes number of zeroes of Pi-function, i.e., extrema or saddlepoints of density.
estimated.roots estimated alpha-values at which extrema or saddlepoints occur.
Author(s)  
Christian Hennig <c.hennig@ucl.ac.uk> \url{http://www.homepages.ucl.ac.uk/~ucakche/}  

References  

Examples  
```r  
q <- piridge.zeros(0.2, c(1, 1), c(2, 5), diag(2), diag(2), by=0.1)  
```

plotcluster  

**Discriminant projection plot.**  

**Description**  
Plots to distinguish given classes by ten available projection methods. Includes classical discriminant coordinates, methods to project differences in mean and covariance structure, asymmetric methods (separation of a homogeneous class from a heterogeneous one), local neighborhood-based methods and methods based on robust covariance matrices. One-dimensional data is plotted against the cluster number.  

**Usage**  
```r  
plotcluster(x, clvecd, clnum=NULL,  
    method=ifelse(is.null(clnum), "dc", "awc"),  
    bw=FALSE,  
    ignorepoints=FALSE, ignorenum=0, pointsbyclvecd=TRUE,  
    xlab=NULL, ylab=NULL,  
    pch=NULL, col=NULL, ...)  
```

**Arguments**  
- `x`  
  the data matrix; a numerical object which can be coerced to a matrix.  
- `clvecd`  
  vector of class numbers which can be coerced into integers; length must equal `nrow(xd)`.  
- `method`  
  one of  
  - "dc" usual discriminant coordinates, see `discrcoord`,  
  - "be" Bhattacharyya coordinates, first coordinate showing mean differences, second showing covariance matrix differences, see `batcoord`,  
  - "vbe" variance dominated Bhattacharyya coordinates, see `batcoord`,  
  - "mvdc" added mean and variance differences optimizing coordinates, see `mvdcoord`,  
  - "ade" asymmetric discriminant coordinates, see `adcoord`,  
  - "awe" asymmetric discriminant coordinates with weighted observations, see `awcoord`,  

"arc" asymmetric discriminant coordinates with weighted observations and robust MCD-covariance matrix, see awcoord,
"nc" neighborhood based coordinates, see ncoord,
"wnc" neighborhood based coordinates with weighted neighborhoods, see ncoord,
"anc" asymmetric neighborhood based coordinates, see ancoord.

Note that "bc", "vbc", "adc", "awc", "arc" and "anc" assume that there are only two classes.

c1num integer. Number of the class which is attempted to plot homogeneously by "asymmetric methods", which are the methods assuming that there are only two classes, as indicated above. c1num is ignored for methods "dc" and "nc".

bw logical. If TRUE, the classes are distinguished by symbols, and the default color is black/white. If FALSE, the classes are distinguished by colors, and the default symbol is pch=1.

ignorepoints logical. If TRUE, points with label ignorenum in clvecd are ignored in the computation for method and are only projected afterwards onto the resulting units. If pch=NULL, the plot symbol for these points is "N".

ignorenum one of the potential values of the components of clvecd. Only has effect if ignorepoints=TRUE, see above.

pointsbyclvecd logical. If TRUE and pch=NULL and/or col=NULL, some hopefully suitable plot symbols (numbers and letters) and colors are chosen to distinguish the values of clvecd, starting with "1"/"black" for the cluster with the smallest clvecd-code (note that colors for clusters with numbers larger than minimum number +3 are drawn at random from all available colors). FALSE produces potentially less reasonable (but nonrandom) standard colors and symbols if method is "dc" or "nc", and will only distinguish whether clvecd=c1num or not for the other methods.

xlab label for x-axis. If NULL, a default text is used.

ylab label for y-axis. If NULL, a default text is used.

pch plotting symbol, see par. If NULL, the default is used.

col plotting color, see par. If NULL, the default is used.

... additional parameters passed to plot or the projection methods.

Note

For some of the asymmetric methods, the area in the plot occupied by the "homogeneous class" (see c1num above) may be very small, and it may make sense to run plotcluster a second time specifying plot parameters xlim and ylim in a suitable way. It often makes sense to magnify the plot region containing the homogeneous class in this way so that its separation from the rest can be seen more clearly.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/
prediction.strength

References


See Also
discrcoord, batcoord, mvdcoord, adcoord, awcoord, ncoord, ancoord.
discrproj is an interface to all these projection methods.
rFace for generation of the example data used below.

Examples

set.seed(4634)
face <- rFace(300,dMoNo=2,dNoEy=8)
grface <- as.integer(attr(face,"grouping"))
plotcluster(face,grface)
plotcluster(face,grface==1)
plotcluster(face,grface, clnum=1, method="vbc")

prediction.strength Prediction strength for estimating number of clusters

Description

Computes the prediction strength of a clustering of a dataset into different numbers of components.
The prediction strength is defined according to Tibshirani and Walther (2005), who recommend to choose as optimal number of cluster the largest number of clusters that leads to a prediction strength above 0.8 or 0.9. See details.

Various clustering methods can be used, see argument clustermethod. In Tibshirani and Walther (2005), only classification to the nearest centroid is discussed, but more methods are offered here, see argument classification.

Usage

prediction.strength(xdata, Gmin=2, Gmax=10, M=50,
clustermethod=kmeansCBI,
classification="centroid",
cutoff=0.8,nnk=1,
distances=inherits(xdata,"dist"),count=FALSE,...)

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

Arguments

**xdata**
data (something that can be coerced into a matrix).

**Gmin**
integer. Minimum number of clusters. Note that the prediction strength for 1 cluster is trivially 1, which is automatically included if Gmin>1. Therefore Gmin<2 is useless.

**Gmax**
integer. Maximum number of clusters.

**M**
integer. Number of times the dataset is divided into two halves.

**clustermethod**
an interface function (the function name, not a string containing the name, has to be provided!). This defines the clustering method. See the "Details"-section of `clusterboot` and `kmeansCBI` for the format. Clustering methods for `prediction.strength` must have a k-argument for the number of clusters, must operate on n times p data matrices and must otherwise follow the specifications in `clusterboot`.

**classification**
string. This determines how non-clustered points are classified to given clusters. Options are explained in `classifnp` and `classifdist`, the latter for dissimilarity data. Certain classification methods are connected to certain clustering methods. classification="averagedist" is recommended for average linkage, classification="centroid" is recommended for k-means, clara and pam, classification="knn" with nnk=1 is recommended for single linkage and classification="qda" is recommended for Gaussian mixtures with flexible covariance matrices.

**cutoff**
numeric between 0 and 1. The optimal number of clusters is the maximum one with prediction strength above cutoff.

**nnk**
number of nearest neighbours if classification="knn", see `classifnp`.

**distances**
logical. If TRUE, data will be interpreted as dissimilarity matrix, passed on to clustering methods as "dist"-object, and `classifdist` will be used for classification.

**count**
logical. TRUE will print current number of clusters and simulation run number on the screen.

**x**
object of class `predstr`.

...arguments to be passed on to the clustering method.

Details

The prediction strength for a certain number of clusters k under a random partition of the dataset in halves A and B is defined as follows. Both halves are clustered with k clusters. Then the points of A are classified to the clusters of B. In the original paper this is done by assigning every observation in A to the closest cluster centroid in B (corresponding to classification="centroid"), but other methods are possible, see `classifnp`. A pair of points A in the same A-cluster is defined to be correctly predicted if both points are classified into the same cluster on B. The same is done with the points of B relative to the clustering on A. The prediction strength for each of the clusterings is the minimum (taken over all clusters) relative frequency of correctly predicted pairs of points of that cluster. The final mean prediction strength statistic is the mean over all 2M clusterings.
Value

prediction.strength gives out an object of class predstr, which is a list with components

- predcorr: list of vectors of length M with relative frequencies of correct predictions (clusterwise minimum). Every list entry refers to a certain number of clusters.
- mean.pred: means of predcorr for all numbers of clusters.
- optimalk: optimal number of clusters.
- cutoff: see above.
- method: a string identifying the clustering method.
- Gmax: see above.
- M: see above.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


See Also

kmeansCBI, classifnp

Examples

```r
options(digits=3)
set.seed(98765)
iriss <- iris[sample(150,20),-5]
prediction.strength(iriss,2,3,M=3)
prediction.strength(iriss,2,3,M=3,clustermethod=claraCBI)
# The examples are fast, but of course M should really be larger.
```

Description

For use within regmix. Generates a random 0-1-matrix with n rows and cln columns so that every row contains exactly one one and every columns contains at least p+3 ones.

Usage

```r
randcmatrix(n,cln,p)
```
Arguments

n positive integer. Number of rows.
cln positive integer. Number of columns.
p positive integer. See above.

Value

An n*cln-matrix.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

See Also

regmix

Examples

set.seed(111)
randcmatrix(10,2,1)

---

randconf Generate a sample indicator vector

Description

Generates a logical vector of length n with p TRUEs.

Usage

randconf(n, p)

Arguments

n positive integer.
p positive integer.

Value

A logical vector.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/
regmix

See Also
sample

Examples
randconf(10,3)

Description
Computes an ML-estimator for clusterwise linear regression under a regression mixture model with Normal errors. Parameters are proportions, regression coefficients and error variances, all independent of the values of the independent variable, and all may differ for different clusters. Computation is by the EM-algorithm. The number of clusters is estimated via the Bayesian Information Criterion (BIC). Note that package flexmix has more sophisticated tools to do the same thing and is recommended. The functions are kept in here only for compatibility reasons.

Usage
regmix(indep, dep, ir=1, nclust=1:7, icrit=1.e-5, minsig=1.e-6, warnings=FALSE)
regem(indep, dep, m, cln, icrit=1.e-5, minsig=1.e-6, warnings=FALSE)

Arguments
indep numerical matrix or vector. Independent variables.
dep numerical vector. Dependent variable.
ir positive integer. Number of iteration runs for every number of clusters.
nclust vector of positive integers. Numbers of clusters.
icrit positive numerical. Stopping criterion for the iterations (difference of loglikelihoods).
mminsig positive numerical. Minimum value for the variance parameters (likelihood is unbounded if variances are allowed to converge to 0).
warnings logical. If TRUE, warnings are given during the EM iteration in case of collinear regressors, too small mixture components and error variances smaller than minimum. In the former two cases, the algorithm is terminated without a result, but an optimal solution is still computed from other algorithm runs (if there are others). In the latter case, the corresponding variance is set to the minimum.
cln positive integer. (Single) number of clusters.
m matrix of positive numericals. Number of columns must be cln. Number of rows must be number of data points. Columns must add up to 1. Initial configuration for the EM iteration in terms of a probability vector for every point which gives its degree of membership to every cluster. As generated by randcmatrix.
Details

The result of the EM iteration depends on the initial configuration, which is generated randomly by `randexmat` for `regmix`. `regmix` calls `regem`. To provide the initial configuration manually, use parameter `m` of `regem` directly. Take a look at the example about how to generate `m` if you want to specify initial parameters.

The original paper DeSarbo and Cron (1988) suggests the AIC for estimating the number of clusters. The use of the BIC is advocated by Wedel and DeSarbo (1995). The BIC is defined here as $2 \times \log \text{lik} - \log(n) \times ((p+3) \times \ln n - 1)$, $p$ being the number of independent variables, i.e., the larger the better.

See the entry for the input parameter `warnings` for the treatment of several numerical problems.

Value

`regmix` returns a list containing the components `clnopt`, `loglik`, `bic`, `coef`, `var`, `eps`, `z`, `g`.

`regem` returns a list containing the components `loglik`, `coef`, `var`, `z`, `g`, `warn`.

- `clnopt`: optimal number of clusters according to the BIC.
- `loglik`: loglikelihood for the optimal model.
- `bic`: vector of BIC values for all numbers of clusters in `nclust`.
- `var`: vector of error variance estimators for the clusters.
- `eps`: vector of cluster proportion estimators.
- `z`: matrix of estimated a posteriori probabilities of the points (rows) to be generated by the clusters (columns). Compare input argument `m`.
- `g`: integer vector of estimated cluster numbers for the points (via argmax over `z`).
- `warn`: logical. `TRUE` if one of the estimated clusters has too few points and/or collinear regressors.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

References


See Also

Regression mixtures can also (and probably better) be computed with the flexmix package, see flexmix. (When I first write the regmix-function, flexmix didn’t exist.)

fixreg for fixed point clusters for clusterwise linear regression.
EMclust for Normal mixture model fitting (non-regression).

Examples

```r
## Not run:
# This apparently gives slightly different
# but data-analytically fine results
# on some versions of R.
set.seed(12234)
data(tonedata)
attach(tonedata)
rm1 <- regmix(stretchratio, tuned, nclust=1:2)
# nclust=1:2 makes the example fast;
# a more serious application would rather use the default.
rm1$g
round(rm1$bic, digits=2)
# start with initial parameter values
cln <- 3
n <- 150
initcoef <- cbind(c(2,0), c(0,1), c(0,2.5))
initvar <- c(0.001, 0.0001, 0.5)
initeps <- c(0.4, 0.3, 0.3)
# computation of m from initial parameters
m <- matrix(nrow=n, ncol=cln)
stm <- numeric(0)
for (i in 1:cln)
  for (j in 1:n){
    m[j,1] <- initeps[i]*dnorm(tuned[j], mean=initcoef[1,i]+initcoef[2,i]*stretchratio[j], sd=sqrt(initvar[i]))
  }
for (j in 1:n){
  stm[j] <- sum(m[j,])
  for (i in 1:cln)
    m[j,1] <- m[j,1]/stm[j]
}
rm2 <- regem(stretchratio, tuned, m, cln)
## End(Not run)
```

**rFace**

"Face-shaped" clustered benchmark datasets

**Description**

Generates "face-shaped" clustered benchmark datasets. This is based on a collaboration with Martin Maechler.
Usage

```r
rFace(n, p = 6, nrep.top = 2, smile.coef = 0.6, dMoNo = 1.2, dNoEy = 1)
```

Arguments

- **n**: integer greater or equal to 10. Number of points.
- **p**: integer greater or equal to 2. Dimension.
- **nrep.top**: integer. Number of repetitions of the hair-top point.
- **smile.coef**: numeric. Coefficient for quadratic term used for generation of mouth-points. Positive values => smile.
- **dMoNo**: number. Distance from mouth to nose.
- **dNoEy**: number. Minimum vertical distance from mouth to eyes.

Details

The function generates a nice benchmark example for cluster analysis. There are six "clusters" in this data, of which the first five are clearly homogeneous patterns, but with different distributional shapes and different qualities of separation. The clusters are distinguished only in the first two dimensions. The attribute grouping is a factor giving the cluster numbers, see below. The sixth group of points corresponds to some hairs, and is rather a collection of outliers than a cluster in itself. This group contains nrep.top+2 points. Of the remaining points, 20% belong to cluster 1, the chin (quadratic function plus noise). 10% belong to cluster 2, the right eye (Gaussian). 30% belong to cluster 3, the mouth (Gaussian/squared Gaussian). 20% belong to cluster 4, the nose (Gaussian/gamma), and 20% belong to cluster 5, the left eye (uniform).

The distributions of the further variables are homogeneous over all points. The third dimension is exponentially distributed, the fourth dimension is Cauchy distributed, all further distributions are Gaussian.

Please consider the source code for exact generation of the clusters.

Value

An n times p numeric matrix with attributes

- **grouping**: a factor giving the cluster memberships of the points.
- **indexlist**: a list of six vectors containing the indices of points belonging to the six groups.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

Examples

```r
set.seed(4634)
f <- rFace(600, dMoNo=2, dNoEy=0)
gf <- as.integer(attr(f, "grouping"))
plot(f, col = gf)
# pairs(f, col = gf, main = "rFace(600,dMoNo=2,dNoEy=0)")
```
Description
Computes \((\alpha \Sigma_1^{-1} + (1-\alpha) \Sigma_2^{-1})^{-1} \cdot \alpha \Sigma_1^{-1} \mu_1 + (1-\alpha) \Sigma_2^{-1} \mu_2\) as required for the computation of the ridgeline (Ray and Lindsay, 2005) to find all density extrema of a two-component Gaussian mixture with mean vectors \(\mu_1\) and \(\mu_2\) and covariance matrices \(\Sigma_1, \Sigma_2\).

Usage
ridgeline(alpha, mu1, mu2, sigma1, sigma2)

Arguments
- alpha: numeric between 0 and 1.
- mu1: mean vector of component 1.
- mu2: mean vector of component 2.
- sigma1: covariance matrix of component 1.
- sigma2: covariance matrix of component 2.

Value
A vector. See above.

Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References

Examples
ridgeline(0.5, c(1,1), c(2,5), diag(2), diag(2))
ridgeline.diagnosis  

Ridgeline plots, ratios and unimodality

Description

Computes ridgeline ratios and unimodality checks for pairs of components given the parameters of a Gaussian mixture. Produces ridgeline plots.

Usage

```r
ridgeline.diagnosis (propvector, muarray, Sigmaarray, 
k=length(propvector), 
ipairs="all", compute.ratio=TRUE, by=0.001, 
ratiocutoff=NULL, ridgelineplot="matrix")
```

Arguments

- `propvector` vector of component proportions. Length must be number of components, and must sum up to 1.
- `muarray` matrix of component means (different components are in different columns).
- `Sigmaarray` three dimensional array with component covariance matrices (the third dimension refers to components).
- `k` integer. Number of components.
- `ipairs` "all" or list of vectors of two integers. If `ipairs="all"`, computations are carried out for all pairs of components. Otherwise, `ipairs` gives the pairs of components for which computations are carried out.
- `compute.ratio` logical. If TRUE, a matrix of ridgeline ratios is computed, see Hennig (2010a).
- `by` real between 0 and 1. Interval width for density computation along the ridgeline.
- `ratiocutoff` real between 0 and 1. If not NULL, the `connection.matrix` (see below) is computed by checking whether ridgeline ratios between components are below `ratiocutoff`.
- `ridgelineplot` one of "none", "matrix", "pairwise". If "matrix", a matrix of pairwise ridgeline plots (see Hennig 2010b) will be plotted. If "pairwise", pairwise ridgeline plots are plotted (you may want to set `par(ask=TRUE)` to see them all). No plotting if "none".

Value

A list with components

- `merged.clusters` vector of integers, stating for every mixture component the number of the cluster of components that would be merged by merging connectivity components of the graph specified by `connection.matrix`.
connection.matrix
zero-one matrix, in which a one means that the mixture of the corresponding pair of components of the original mixture is either unimodel (if ratiocutoff=NULL) or that their ridgeline ratio is above ratiocutoff. If ipairs="all", ignored pairs always have 0 in this matrix, same for ratio.matrix.

ratio.matrix
matrix with entries between 0 und 1, giving the ridgeline ratio, which is the density minimum of the mixture of the corresponding pair of components along the ridgeline divided by the minimum of the two maxima closest to the beginning and the end of the ridgeline.

Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References
Hennig, C. (2010a) Methods for merging Gaussian mixture components, Advances in Data Analysis and Classification, 4, 3-34.

See Also
ridgeline, dridgeline, piridge, piridge.zeros

Examples
muarray <- cbind(c(0, 0), c(0, 0.1), c(10, 10))
sigmaarray <- array(c(diag(2), diag(2), diag(2)), dim=c(2, 2, 3))
rd <-
ridgeline.diagnosis(c(0.5, 0.3, 0.2), muarray, sigmaarray, ridgelineplot="matrix", by=0.1)
# Much slower but more precise with default by=0.001.

simmatrix
Extracting intersections between clusters from fpc-object

Description
Extracts the information about the size of the intersections between representative Fixed Point Clusters (FPCs) of stable groups from the output of the FPC-functions fixreg and fixmahal.

Usage
simmatrix(fpcobj)
solvecov

Arguments

fpcobj  
an object of class rfpc or mfpc.

Value

A non-negative real-valued vector giving the number of points in the intersections of the representative FPCs of stable groups.

Note

The intersection between representative FPCs no. i and j is at position sseg(i, j).

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk>  http://www.homepages.ucl.ac.uk/~ucakche/

See Also

fixmahal, fixreg, sseg

Examples

```
set.seed(190000)
data(tonedata)  
# Note: If you do not use the installed package, replace this by  
# tonedata <- read.table("(path/)tonedata.txt", header=TRUE)  
attach(tonedata)  
tonefix <- fixreg(stretchratio, tuned, mtf=1, ir=20)  
simmatrix(tonefix)[sseg(2,3)]
```

---

solvecov

**Inversion of (possibly singular) symmetric matrices**

Description

Tries to invert a matrix by solve. If this fails because of singularity, an eigenvector decomposition is computed, and eigenvalues below 1/cmax are replaced by 1/cmax, i.e., cmax will be the corresponding eigenvalue of the inverted matrix.

Usage

```
solvecov(m, cmax = 1e+10)
```

Arguments

- **m**: a numeric symmetric matrix.
- **cmax**: a positive value, see above.
Value
A list with the following components:

inv       the inverted matrix
coll      TRUE if solve failed because of singularity.

Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

See Also
solve, eigen

Examples
x <- c(1,0,0,1,0,1,0,0,1)
dim(x) <- c(3,3)
solvecov(x)

sseg(i, j) gives the position of the similarity of objects i and j in the similarity vectors produced by fixreg and fixmahal. sseg should only be used as an auxiliary function in fixreg and fixmahal.

Usage
sseg(i, j)

Arguments
i     positive integer.
j     positive integer.

Value
A positive integer.

Author(s)
Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

Examples
sseg(3,4)
tdecomp

Root of singularity-corrected eigenvalue decomposition

Description

Computes transposed eigenvectors of matrix \( m \) times diagonal of square root of eigenvalues so that eigenvalues smaller than 1e-6 are set to 1e-6.

Usage

tdecomp(m)

Arguments

\( m \) a symmetric matrix of minimum format 2*2.

Details

Thought for use in discrcoord only.

Value

a matrix.

Note

Thought for use within discrcoord only.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

Examples

\[
x <- rnorm(10) 
y <- rnorm(10) 
z <- cov(cbind(x,y)) 
round(tdecomp(z), digits=2)
\]
The tone perception data stem from an experiment of Cohen (1980) and have been analyzed in de Veaux (1989). A pure fundamental tone was played to a trained musician. Electronically generated overtones were added, determined by a stretching ratio of stretchratio. stretchratio=2.0 corresponds to the harmonic pattern usually heard in traditional definite pitched instruments. The musician was asked to tune an adjustable tone to the octave above the fundamental tone. tuned gives the ratio of the adjusted tone to the fundamental, i.e. tuned=2.0 would be the correct tuning for all stretchratio-values. The data analyzed here belong to 150 trials with the same musician. In the original study, there were four further musicians.

Usage
data(tonedata)

Format
A data frame with 2 variables stretchratio and tuned and 150 cases.

Source

References

Description
Checks whether a series of fitted density values (such as given out as y-component of density) is unimodal.

Usage
unimodal.ind(y)
weightplots

Arguments

- `y` numeric vector of fitted density values in order of increasing x-values such as given out as y-component of `density`.

Value

Logical. TRUE if unimodal.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

Examples

```r
unimodal.ind(c(1,3,3,4,2,1,0,0))
```

---

**Ordered posterior plots**

Description

Ordered posterior plots for Gaussian mixture components, see Hennig (2010).

Usage

```r
weightplots(z, clusternumbers="all", clustercol=2,
           allcol=grey(0.2+((1:ncol(z))-1)*
                 0.6/(ncol(z)-1)),
           lty=rep(1,ncol(z)),clusterlwd=3,
           legendposition="none",
           weightcutoff=0.01,ask=TRUE, ...)
```

Arguments

- `z` matrix with rows corresponding to observations and columns corresponding to mixture components. Entries are probabilities that an observation has been generated by a mixture component. These will normally be estimated a posteriori probabilities, as generated as component `z` of the output object from `summary.mclustBIC`.
- `clusternumbers` "all" or vector of integers. Numbers of components for which plots are drawn.
- `clustercol` colour used for the main components for which a plot is drawn.
- `allcol` colours used for respective other components in plots in which they are not main components.
- `lty` line types for components.
- `clusterlwd` numeric. Line width for main component.
legendposition  "none" or vector with two coordinates in the plot, where a legend should be printed.
weightcutoff   numeric between 0 and 1. Observations are only taken into account for which the posterior probability for the main component is larger than this.
ask           logical. If TRUE, it sets par(ask=TRUE) in the beginning and par(ask=FALSE) after all plots were showed.

Details

Shows posterior probabilities for observations belonging to all mixture components on the y-axis, with points ordered by posterior probability for main component.

Value

Invisible matrix of posterior probabilities z from mclustsummary.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

References


Examples

```r
require(mclust)
require(MASS)
data(crabs)
dc <- crabs[,4:8]
cm <- mclustBIC(crabs[,4:8],G=9,modelNames="EEE")
scm <- summary(cm,crabs[,4:8])
weightplots(scm$z,clusternumbers=1:3,ask=FALSE)
weightplots(scm$z,clusternumbers=1:3,allcol=1:9, ask=FALSE, legendposition=c(5,0.7))
# Remove ask=FALSE to have time to watch the plots.
```

---

**wfu**  

*Weight function (for Mahalanobis distances)*

**Description**

Function of the elements of md, which is 1 for arguments smaller than ca, 0 for arguments larger than ca2 and linear (default: continuous) in between.

Thought for use in fixmahal.
Usage

\[ wfu(md, ca, ca2, a1 = 1/(ca - ca2), a0 = -a1 * ca2) \]

Arguments

- \( md \) vector of positive numericals.
- \( ca \) positive numerical.
- \( ca2 \) positive numerical.
- \( a1 \) numerical. Slope.
- \( a0 \) numerical. Intercept.

Value

A vector of numericals between 0 and 1.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

See Also

fixmahal

Examples

```r
md <- seq(0,10,by=0.1)
round(wfu(md,ca=5,ca2=8),digits=2)
```

---

**xtable**

*Partition crosstable with empty clusters*

Description

This produces a crosstable between two integer vectors (partitions) of the same length with a given maximum vector entry \( k \) so that the size of the table is \( k \times k \) with zeroes for missing entries between 1 and \( k \) (the command `table` does pretty much the same thing but will leave out missing entries).

Usage

\[ xtable(c1,c2,k) \]

Arguments

- \( c1 \) vector of integers.
- \( c2 \) vector of integers of same length as \( c1 \).
- \( k \) integer. Must be larger or equal to maximum entry in \( c1 \) and \( c2 \).
zmisclassification.matrix

Value

A matrix of dimensions \(c(k, k)\). Entry \([i, j]\) gives the number of places in which \(c_1 = i \land c_2 = j\).

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> http://www.homepages.ucl.ac.uk/~ucakche/

See Also

table

Examples

c1 <- 1:3
c2 <- c(1,1,2)
xtab(c1,c2,3)

zmisclassification.matrix

Matrix of misclassification probabilities between mixture components

Description

Matrix of misclassification probabilities in a mixture distribution between two mixture components from estimated posterior probabilities regardless of component parameters, see Hennig (2010).

Usage

zmisclassification.matrix(z, pro=NULL, clustering=NULL, ipairs=NULL, symmetric=TRUE, stat=max)

Arguments

- **z**: matrix of posterior probabilities for observations (rows) to belong to mixture components (columns), so entries need to sum up to 1 for each row.
- **pro**: vector of component proportions, need to sum up to 1. Computed from \(z\) as default.
- **clustering**: vector of integers giving the estimated mixture components for every observation. Computed from \(z\) as default.
- **ipairs**: "all" or list of vectors of two integers. If \(ipairs="all"\), computations are carried out for all pairs of components. Otherwise, \(ipairs\) gives the pairs of components for which computations are carried out.
- **symmetric**: logical. If TRUE, the matrix is symmetrised, see parameter \(stat\).
- **stat**: "max" or "mean". The statistic by which the two misclassification probabilities are aggregated if \(symmetric=TRUE\).
Value

A matrix with the (symmetrised, if required) misclassification probabilities between each pair of mixture components. If `symmetric=FALSE`, matrix entry \([i,j]\) is the estimated probability that an observation generated by component \(j\) is classified to component \(i\) by maximum a posteriori rule.

Author(s)

Christian Hennig <c.hennig@ucl.ac.uk> [http://www.homepages.ucl.ac.uk/~ucakche/](http://www.homepages.ucl.ac.uk/~ucakche/)

References


See Also

`confusion`

Examples

```r
set.seed(12345)
m <- rpois(20, lambda=5)
dim(m) <- c(5,4)
m <- m/apply(m,1,sum)
round(zmisclassification.matrix(m,symmetric=FALSE),digits=2)
```
Index

*Topic arith
  can, 15
cweight, 39
wfu, 121
*Topic array
  con.comp, 35
  solvecov, 116
tdecomp, 118
xtable, 122
*Topic classif
  adcoord, 6
  ancoord, 7
  awcoord, 8
  batcoord, 10
discrcoord, 44
discrproj, 47
mvdcoord, 94
ncoord, 95
plotcluster, 103
*Topic cluster
  bhattacharyya.matrix, 13
calinhara, 14
cdbw, 17
classifdist, 18
clucols, 20
clujaccard, 21
cludepect, 21
cluster.stats, 22
cluster.varstats, 26
clusterboot, 28
cmahal, 34
con.comp, 35
confusion, 36
cvnn, 38
dbscan, 40
dipp.tantrum, 42
diptest.multi, 43
distancefactor, 49
distcritmulti, 50
dridgeline, 52
dudahart2, 53
extract.mixturepars, 54
findrep, 55
fixmahal, 56
fixreg, 62
flexmixedruns, 68
fpclusters, 71
itnumber, 71
kmeansCBI, 73
kmeansruns, 78
lcmixed, 80
mahalconf, 85
mergenormals, 86
mergeparameters, 89
minsize, 91
mixdens, 92
mixpredictive, 93
neginc, 97
nselectboot, 98
pamk, 99
piridge, 101
piridge.zeros, 102
prediction.strength, 105
randcmatrix, 107
regmix, 109
ridgeline, 113
ridgeline.diagnosis, 114
weightplots, 120
zmisclassification.matrix, 123
*Topic datasets
  tonedata, 119
*Topic data
  rface, 111
*Topic distribution
  randconf, 108
*Topic manip
  cat2bin, 16
discrete.recode, 45
jittervar, 72
*Topic multivariate
    adcoord, 6
    ancoord, 7
    awcoord, 8
    batcoord, 10
    bhattacharyya.dist, 12
    bhattacharyya.matrix, 13
    classifdist, 18
    cluster.stats, 22
    clusterboot, 28
    confusion, 36
    cov.wml, 37
    dbscan, 40
    dipetest.multi, 43
    discrcoord, 44
    discrproj, 47
    dridgeline, 52
    extract.mixturepars, 54
    fixmahal, 56
    kmeansCBI, 73
    kmeansruns, 78
    localshape, 82
    mahalanodisc, 83
    mahalanofix, 84
    mahalconf, 85
    mergenormals, 86
    mergeparameters, 89
    mixdens, 92
    mixpredictive, 93
    mvdcoord, 94
    ncoord, 95
    nselectboot, 98
    pamk, 99
    piridge, 101
    piridge.zeros, 102
    plotcluster, 103
    prediction.strength, 105
    ridgeline, 113
    ridgeline.diagnosis, 114
    weightplots, 120
    zmisclassification.matrix, 123
*Topic regression
    fixreg, 62
    regmix, 109
*Topic robust
    fixmahal, 56
    fixreg, 62
*Topic univar
    clusexpect, 21
    itnumber, 71
    minsize, 91
    unimodal.ind, 119
*Topic utilities
    simmatrix, 115
    sseg, 117
    adcoord, 6, 47, 48, 103, 105
    ancoord, 7, 47, 48, 104, 105
    awcoord, 8, 40, 47, 48, 83, 103–105
    batcoord, 10, 45, 47, 48, 59, 61, 103, 105
    batvarcoord (batcoord), 10
    bhattacharyya.dist, 12, 14
    bhattacharyya.matrix, 13, 87
    calinhara, 4, 14, 26, 78, 80, 99
    can, 15, 63, 67
    cat2bin, 16
    cdbw, 17, 55, 56
    clara, 5, 32, 76–78, 99–101
    claraCBI, 33
    claraCBI (kmeansCBI), 73
    classifdist, 18, 98, 99, 106
    classifnp, 98, 99, 106, 107
    classifnp (classifdist), 18
    clinclus, 20
    clugrey (clucols), 20
    clujaccard, 21
    clusexpect, 21, 65, 67, 72, 91, 92
    cluster.stats, 15, 22, 51, 53
    cluster.varstats, 26
    clusterboot, 26, 28, 73, 76, 78, 98, 99, 106
    clusym (clucols), 20
    cmahal, 34, 57, 58, 61
    cmdscale, 75
    con.comp, 35
    confusion, 36, 124
    cov, 38
    cov.rob, 7, 9, 57, 61, 84, 85, 94, 96
    cov.wml, 37, 61
    cov.wt, 37, 38
    covMcd, 82
    cutree, 36
    cvnn, 38
    cweight, 39
    daisy, 50, 51
INDEX

data.matrix, 46
dbscan, 32, 40, 76–78
dbscanCBI, 33
dbscanCBI (kmeansCBI), 73
density, 43, 119, 120
dip, 42, 43
dip.test, 42, 43, 87
dipp.tantrum, 42, 88
dip.test.multi, 43
disccritmulti, 11, 12, 44, 47, 48, 103, 105, 118
discrete.recode, 16, 45, 70, 81
discretpair, 7, 8, 10, 26, 27, 47, 95, 96, 105
dist, 26, 33, 50, 51, 78
distancefactor, 49
distcritmulti, 25, 26, 50, 100, 101
distclustCBI, 33
distclustCBI (kmeansCBI), 73
distclusttreeCBI (kmeansCBI), 73
distnoiseclustCBI, 33
distnoiseclustCBI (kmeansCBI), 73
disttrimkmeansCBI, 33
disttrimkmeansCBI (kmeansCBI), 73
drigeline, 42, 43, 73
findrep, 55
fixreg, 15, 16, 21, 22, 61, 62, 71, 72, 91, 92, 111, 115, 116
fixflexmix, 4, 70, 80, 81, 111
flexmixedruns, 68, 80, 81
fpc-package, 3
fpclusters, 71
fpclusters.mfpc (fixmahal), 56
fpclusters.rfpc (fixreg), 62
fpmi (fixmahal), 56
grey, 20
hc, 75
hclust, 36, 75, 76, 78
hclustCBI, 33
hclustCBI (kmeansCBI), 73
hclusttreeCBI, 33
hclusttreeCBI (kmeansCBI), 73
isoMDS, 75
itnumber, 64, 67, 71, 92
jitter, 73
jittervar, 72
kmeans, 5, 31, 76, 78–80
kmeansCBI, 5, 31–33, 73, 98, 99, 106, 107
kmeansruns, 76, 78, 78
knn, 19
lcmixed, 46, 50, 70, 80
lda, 19
legend, 121
localshape, 82
mahalanodisc, 83
mahalanofix, 84
mahalanofuz (mahalanofix), 84
mahalCBI, 33
mahalconf, 57, 59, 61, 85
mclustBIC, 30, 31, 54, 75, 76, 78, 86, 88
mclustModelNames, 92
mergenormals, 32, 76, 77, 86, 93, 94
mergenormCBI (kmeansCBI), 73
mergeparameters, 88, 89
minsize, 67, 91
mixdens, 92
mixpredictive, 88, 93
mvdcoord, 47, 48, 94, 103, 105
ncoord, 47, 48, 95, 104, 105
neginc, 97
NNClean, 75, 78
noisemclustCBI, 33
noisemclustCBI (kmeansCBI), 73
nselectboot, 19, 98
pam, 5, 19, 32, 50, 76–78, 99–101
pam.object, 78, 99
pamk, 32, 77, 78, 80, 99
pamkCBI, 33
pamkCBI (kmeansCBI), 73
par, 58, 59, 65, 104
pdfclustCBI (kmeansCBI), 73
pdfCluster, 77, 78
piridge, 101, 115
piridge.zeros, 102, 115
plot.clboot (clusterboot), 28
plot.dbscan (dbscan), 40
plot.mfpc (fixmahal), 56
plot.rfpc (fixreg), 62
plotcluster, 7, 8, 10, 12, 31, 45, 61, 95, 96, 103
predict.dbscan (dbscan), 40
prediction.strength, 19, 94, 105
print.clboot (clusterboot), 28
print.dbscan (dbscan), 40
print.mfpc (fixmahal), 56
print.predstr (prediction.strength), 105
print.rfpc (fixreg), 62
print.summary.mergenorm (mergenormals), 86
print.summary.mfpc (fixmahal), 56
print.summary.rfpc (fixreg), 62
print.varwisetables (cluster.varstats), 26

tdecomp, 6, 8, 9, 44, 118
tonedata, 119
trimkmeans, 32, 76–78
trimkmeansCBI, 73
trimkmeansCBI (kmeansCBI), 73
try, 69
unimodal.ind, 119
var, 38
weightplots, 120
wfus, 57, 59, 61, 121
xtable, 122
zmisclassification.matrix, 123

qda, 19

randcmatrix, 107, 109, 110
randconf, 108
regem (regmix), 109
regmix, 67, 108, 109
rFace, 7, 8, 10, 12, 45, 48, 61, 95, 96, 105, 111
rfpi (fixreg), 62
ridgeline, 113, 115
ridgeline.diagnosis, 87, 114

sammon, 75
sample, 109
silhouette, 24, 26, 51
simmatrix, 115
solve, 117
solvecov, 59, 61, 83–86, 116
specc, 32, 77
speccCBI (kmeansCBI), 73
ssseg, 60, 61, 66, 67, 116, 117
summary.mclustBIC, 54, 75, 86, 92, 120
summary.mergenorm (mergenormals), 86
summary.mfpc (fixmahal), 56
summary.rfpc (fixreg), 62

table, 122, 123
tclust, 78
tclustCBI (kmeansCBI), 73