Package ‘tclust’

July 22, 2025

Type Package

Encoding UTF-8

Title Robust Trimmed Clustering

Version 2.1-2

VersionNote Released 2.1-0 on 2025-02-19 on CRAN

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

Description Provides functions for robust trimmed clustering. The methods are described in Garcia-Escudero (2008) <doi:10.1214/07-AOS515>, Fritz et al. (2012) <doi:10.18637/jss.v047.i12>, Garcia-Escudero et al. (2011) <doi:10.1007/s11222-010-9194-z> and others.

Depends R(>= 3.6.2)

Imports Rcpp (>= 1.0.7), doParallel, parallel, foreach, MASS, rlang

Suggests mclust, cluster, sn

LazyLoad yes

License GPL-3

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

RoxygenNote 7.3.2

URL `https://github.com/valentint/tclust`

BugReports `https://github.com/valentint/tclust/issues`

Repository CRAN

Author Valentin Todorov [aut, cre] (ORCID: <`https://orcid.org/0000-0003-4215-0245`>), Luis Angel García Escudero [aut], Agustín Mayo Iscar [aut], Javier Crespo Guerrero [aut], Heinrich Fritz [aut]

Date/Publication 2025-06-29 14:40:02 UTC

Contents

`ctlcurves` *Classification Trimmed Likelihood Curves*

Description

The function applies `tclust` several times on a given dataset while parameters `alpha` and `k` are altered. The resulting object gives an idea of the optimal trimming level and number of clusters considering a particular dataset.

Usage

```
ctlcurves(
  x,
  k = 1:4,
  alpha = seq(0, 0.2, len = 6),
  restr.fact = 50,
  parallel = FALSE,
  trace = 1,
  ...
)
```

Arguments

`x`	A matrix or data frame of dimension n x p, containing the observations (row-wise).
`k`	A vector of cluster numbers to be checked. By default cluster numbers from 1 to 5 are examined.
`alpha`	A vector containing the alpha levels to be checked. By default `alpha` levels from 0 to 0.2 (continuously increased by 0.01), are checked.
`restr.fact`	The restriction factor passed to `tclust`.
`parallel`	A logical value, to be passed further to `tclust()`.
`trace`	Defines the tracing level, which is set to `1` by default. Tracing level `2` gives additional information on the current iteration.
`...`	Further arguments (as e.g. `restr`), passed to `tclust`

Details

These curves show the values of the trimmed classification (log-)likelihoods when altering the trimming proportion `alpha` and the number of clusters `k`. The careful examination of these curves provides valuable information for choosing these parameters in a clustering problem. For instance, an appropriate `k` to be chosen is one that we do not observe a clear increase in the trimmed classification likelihood curve for `k` with respect to the `k+1` curve for almost all the range of alpha values. Moreover, an appropriate choice of parameter alpha may be derived by determining where an initial fast increase of the trimmed classification likelihood curve stops for the final chosen `k`. A more detailed explanation can be found in García-Escudero et al. (2011).

Value

The function returns an S3 object of type `ctlcurves` containing the following components:

- `par` A list containing all the parameters passed to this function
- `obj` An array containing the objective functions values of each computed cluster-solution
- `min.weights` An array containing the minimum cluster weight of each computed cluster-solution

References

García-Escudero, L.A.; Gordaliza, A.; Matrán, C. and Mayo-Iscar, A. (2011), "Exploring the number of groups in robust model-based clustering." *Statistics and Computing*, **21** pp. 585-599, <doi:10.1007/s11222-010-9194-z>

Examples

```
#--- EXAMPLE 1 ------------------------------------------

sig <- diag (2)
cen <- rep (1, 2)
```

```
x <- rbind(MASS::mvrnorm(108, cen * 0,   sig),
        MASS::mvrnorm(162, cen * 5,   sig * 6 - 2),
        MASS::mvrnorm(30, cen * 2.5, sig * 50))

ctl <- ctlcurves(x, k = 1:4)
ctl

  ##  ctl-curves
plot(ctl)  ##  --> selecting k = 2, alpha = 0.08

  ##  the selected model
plot(tclust(x, k = 2, alpha = 0.08, restr.fact = 7))

#--- EXAMPLE 2 ------------------------------------------

data(geyser2)
ctl <- ctlcurves(geyser2, k = 1:5)
ctl

  ##  ctl-curves
plot(ctl)  ##  --> selecting k = 3, alpha = 0.08

  ##  the selected model
plot(tclust(geyser2, k = 3, alpha = 0.08, restr.fact = 5))


#--- EXAMPLE 3 ------------------------------------------

data(swissbank)
ctl <- ctlcurves(swissbank, k = 1:5, alpha = seq (0, 0.3, by = 0.025))
ctl

  ##  ctl-curves
plot(ctl)  ##  --> selecting k = 2, alpha = 0.1

  ##  the selected model
plot(tclust(swissbank, k = 2, alpha = 0.1, restr.fact = 50))
```

DiscrFact *Discriminant Factor analysis for* `tclust` *objects*

Description

Analyzes a `tclust`-object by calculating discriminant factors and comparing the quality of the actual cluster assignments to that of the second best possible assignment for each observation. Cluster assignments of observations with large discriminant factors are considered "doubtful" decisions. Silhouette plots give a graphical overview of the discriminant factors distribution (see `plot.DiscrFact`). More details can be found in García-Escudero et al. (2011).

Usage

```
DiscrFact(x, threshold = 1/10)
```

Arguments

`x`	A `tclust` object.
`threshold`	A cluster assignment or a trimming decision for an observation with a discriminant factor larger than `log(threshold)` is considered a "doubtful" decision.

Value

The function returns an S3 object of type `DiscrFact` containing the following components:

- `x` A `tclust` object.
- `ylimmin` A minimum y-limit calculated for plotting purposes.
- `ind` The actual cluster assignment.
- `ind2` The second most likely cluster assignment for each observation.
- `lik` The (weighted) likelihood of the actual cluster assignment of each observation.
- `lik2` The (weighted) likelihood of the second best cluster assignment of each observation.
- `assignfact` The factor `log(disc/disc2)`.
- `threshold` The threshold used for deciding whether `assignfact` indicates a "doubtful" assignment.
- `mean.DiscrFact` A vector of length `k + 1` containing the mean discriminant factors for each cluster (including the outliers).

References

García-Escudero, L.A.; Gordaliza, A.; Matrán, C. and Mayo-Iscar, A. (2011), "Exploring the number of groups in robust model-based clustering." *Statistics and Computing*, **21** pp. 585-599, <doi:10.1007/s11222-010-9194-z>

`estepRR` *Function to perform the E-step for a Gaussian mixture distribution*

Description

Compute the log PDF for each observation, the posterior probabilities and the objective function (total log-likelihood) for a Gaussian mixture distribution

Arguments

`ll`	Rcpp::NumericMatrix, n-by-k where `n` is the number of observations and `k` is the number of clusters.

Details

Formally a mixture model corresponds to the mixture distribution that represents the probability distribution of observations in the overall population. Mixture models are used to make statistical inferences about the properties of the sub-populations given only observations on the pooled population, without sub-population-identity information. Mixture modeling approaches assume that data at hand $y_1, ..., y_n$ in R^p come from a probability distribution with density given by the sum of k components

$$\sum_{j=1}^{k} \pi_j \phi(\cdot, \theta_j)$$

with $\phi(\cdot, \theta_j)$ being the p-variate (generally multivariate normal) densities with parameters θ_j, $j = 1, \dots, k$. Generally $\theta_j = (\mu_j, \Sigma_j)$ where μ_j is the population mean and Σ_j is the covariance matrix for component j. π_j is the (prior) probability of component j. The objective function is obj is equal to

$$obj = \log \left(\prod_{i=1}^{n} \sum_{j=1}^{k} \pi_j \phi(y_i;\ \theta_j) \right)$$

or

$$obj = \sum_{i=1}^{n} \log \left(\sum_{j=1}^{k} \pi_j \phi(y_i;\ \theta_j) \right)$$

where k is the number of components of the mixture and π_j are the component probabilitites and θ_j are the parameters of the j-th mixture component.

Value

The function returns a list with the following elements:

- obj The value of the objective function (total log-likelihood)
- postprob an `n-by-k` matrix with the posterior probablilities
- logpdf a vector of length `n` containing the log PDF for each observation

References

McLachlan, G.J.; Peel, D. (2000). Finite Mixture Models. Wiley. ISBN 0-471-00626-2

Examples

```
##      Generate two Gaussian normal distributions
##      and do not produce plots

         mu1 = c(1,2)
         sigma1 = matrix(c(2, 0, 0, .5), nrow=2, byrow=TRUE)    #[2 0; 0 .5];
         mu2 = c(-3, -5)
         sigma2 = matrix(c(1, 0, 0, 1), nrow=2, byrow=TRUE)
         n1 = 100
         n2 = 200
```

```
Y = rbind(MASS::mvrnorm(n1, mu1, sigma1),
          MASS::mvrnorm(n2, mu2, sigma2))
k = 2
pi = c(1/3, 2/3)
mu = rbind(mu1, mu2)
sigma = array(0, dim=c(2,2,2))
sigma[,,1] = sigma1
sigma[,,2] = sigma2

ll = matrix(0, nrow=n1+n2, ncol=2)
for(j in 1:k)
    ll[,j] = log(pi[j]) +  tclust:::dmvnrm(Y, mu[j,], sigma[,,j])

dd = tclust:::estepRR(ll)
dd$obj
dd$logpdf
dd$postprob
```

`flea` *Flea*

Description

Flea-beetle measurements

Usage

```
data(flea)
```

Format

A data frame with 74 rows and 7 variables: six explanatory and one response variable - `species`. The variables are as follows:

- tars1: width of the first joint of the first tarsus in microns (the sum of measurements for both tarsi)
- tars2: the same for the second joint
- head: the maximal width of the head between the external edges of the eyes in 0.01 mm
- ade1: the maximal width of the aedeagus in the fore-part in microns
- ade2: the front angle of the aedeagus (1 unit = 7.5 degrees)
- ade3: the aedeagus width from the side in microns
- species, which species is being examined - `Concinna`, `Heptapotamica`, `Heikertingeri`

References

A. A. Lubischew (1962), On the Use of Discriminant Functions in Taxonomy, *Biometrics*, **18**4 pp.455–477.

Examples

```
data(flea)
head(flea)
```

`geyser2` *Old Faithful Geyser Data*

Description

A bivariate data set obtained from the Old Faithful Geyser, containing the eruption length and the length of the previous eruption for 271 eruptions of this geyser in minutes.

Usage

```
data(geyser2)
```

Format

A data frame containing 272 observations in 2 variables. The variables are as follows:

- `Eruption length` The eruption length in minutes.
- `Previous eruption length` The length of the previous eruption in minutes.

Source

This particular data structure can be obtained by applying the following code to the "Old Faithful Geyser" (`faithful` data set (Härdle 1991) in the package `datasets`):

```
f1 <- faithful[,1]
geyser2 <- cbind(f1[-length(f1)], f1[-1])
colnames(geyser2) <- c("Eruption length",
"Previous eruption length")
```

References

García-Escudero, L.A. and Gordaliza, A. (1999). Robustness properties of k-means and trimmed k-means. *Journal of the American Statistical Assoc.*, Vol.94, No.447, 956–969.

Härdle, W. (1991). *Smoothing Techniques with Implementation in S.*, New York: Springer.

LG5data *LG5data data*

Description

A data set in dimension 10 with three clusters around affine subspaces of common intrinsic dimension. A 10% background noise is added uniformly distributed in a rectangle containing the three main clusters.

Usage

```
data(LG5data)
```

Format

The first 10 columns are the variables. The last column is the true classification vector where symbol "0" stands for the contaminating data points.

Examples

```
#--- EXAMPLE 1 ------------------------------------------
data (LG5data)
x <- LG5data[, 1:10]
clus <- rlg(x, d = c(2,2,2), alpha=0.1, trace=TRUE)
plot(x, col=clus$cluster+1)
```

M5data *M5data data*

Description

A bivariate data set obtained from three normal bivariate distributions with different scales and proportions 1:2:2. One of the components is very overlapped with another one. A 10% background noise is added uniformly distributed in a rectangle containing the three normal components and not very overlapped with the three mixture components. A precise description of the M5 data set can be found in García-Escudero et al. (2008).

Usage

```
data(M5data)
```

Format

The first two columns are the two variables. The last column is the true classification vector where symbol "0" stands for the contaminating data points.

Source

García-Escudero, L.A.; Gordaliza, A.; Matrán, C. and Mayo-Iscar, A. (2008), "A General Trimming Approach to Robust Cluster Analysis". Annals of Statistics, Vol.36, pp. 1324-1345.

Examples

```
#--- EXAMPLE 1 ------------------------------------------
data (M5data)
x <- M5data[, 1:2]
clus <- tclust(x, k=3, alpha=0.1, nstart=200, niter1=3, niter2=17,
   nkeep=10, opt="HARD", equal.weights=FALSE, restr.fact=50, trace=TRUE)
plot (x, col=clus$cluster+1)
```

---

pine *Pinus nigra dataset*

Description

To study the growth of the wood mass in a cultivated forest of *Pinus nigra* located in the north of Palencia (Spain), a sample of 362 trees was studied. The data set is made of measurements of heights (in meters), in variable "HT", and diameters (in millimetres), in variable "Diameter", of these trees. The presence of three linear groups can be guessed apart from a small group of trees forming its own cluster with larger heights and diameters one isolated tree with the largest diameter but small height. More details on the interpretation of this dataset in García-Escudero et al (2010).

Usage

```
data(pine)
```

Format

A data frame containing 362 observations in 2 variables. The variables are as follows:

- `Diameter` Diameter
- `HT` Height

References

García-Escudero, L. A., Gordaliza, A., Mayo-Iscar, A., and San Martín, R. (2010). Robust clusterwise linear regression through trimming. *Computational Statistics & Data Analysis*, 54(12), 3057–3069.

plot.ctlcurves — *The* plot *method for objects of class* ctlcurves

Description

The plot method for class ctlcurves: This function implements a series of plots, which display characteristic values of the each model, computed with different values for k and alpha.

Usage

```
## S3 method for class 'ctlcurves'
plot(
  x,
  what = c("obj", "min.weights", "doubtful"),
  main,
  xlab,
  ylab,
  xlim,
  ylim,
  col,
  lty = 1,
  ...
)
```

Arguments

x	The ctlcurves object to be shown
what	A string indicating which type of plot shall be drawn. See the details section for more information.
main	A character-string containing the title of the plot.
xlab, ylab, xlim, ylim	Arguments passed to plot().
col	A single value or vector of line colors passed to lines.
lty	A single value or vector of line colors passed to lines.
...	Arguments to be passed to or from other methods.

Details

These curves show the values of the trimmed classification (log-)likelihoods when altering the trimming proportion alpha and the number of clusters k. The careful examination of these curves provides valuable information for choosing these parameters in a clustering problem. For instance, an appropriate k to be chosen is one that we do not observe a clear increase in the trimmed classification likelihood curve for k with respect to the k+1 curve for almost all the range of alpha values. Moreover, an appropriate choice of parameter alpha may be derived by determining where an initial fast increase of the trimmed classification likelihood curve stops for the final chosen k. A more detailed explanation can be found in García-Escudero et al. (2011).

This function implements a series of plots, which display characteristic values of the each model, computed with different values for `k` and `alpha`.

`"obj"` Objective function values.

`"min.weights"` The minimum cluster weight found for each computed model. This plot is intended to spot spurious clusters, which in general yield quite small weights.

`"doubtful"` The number of "doubtful" decisions identified by `DiscrFact`.

References

García-Escudero, L.A.; Gordaliza, A.; Matrán, C. and Mayo-Iscar, A. (2011), "Exploring the number of groups in robust model-based clustering." *Statistics and Computing*, **21** pp. 585-599, <doi:10.1007/s11222-010-9194-z>

Examples

```
#--- EXAMPLE 1 ------------------------------------------


sig <- diag (2)
cen <- rep (1, 2)
x <- rbind(MASS::mvrnorm(108, cen * 0,   sig),
        MASS::mvrnorm(162, cen * 5,   sig * 6 - 2),
        MASS::mvrnorm(30, cen * 2.5, sig * 50))

(ctl <- ctlcurves(x, k = 1:4))

plot(ctl)
```

`plot.DiscrFact` *The* `plot` *method for objects of class* `DiscrFact`

Description

The `plot` method for class `DiscrFact`: Next to a plot of the `tclust` object which has been used for creating the `DiscrFact` object, a silhouette plot indicates the presence of groups with a large amount of doubtfully assigned observations. A third plot similar to the standard `tclust` plot serves to highlight the identified doubtful observations.

Usage

```
## S3 method for class 'DiscrFact'
plot(
  x,
  enum.plots = FALSE,
  xlab = "Discriminant Factor",
  ylab = "Clusters",
  print.DiscrFact = TRUE,
  xlim,
  col.nodoubt = grey(0.8),
  by.cluster = FALSE,
  ...
)
```

Arguments

`x`	An object of class `DiscrFact` as returned from DiscrFact()
`enum.plots`	A logical value indicating whether the plots shall be enumerated in their title ("(a)", "(b)", "(c)").
`xlab`, `ylab`, `xlim`	Arguments passed to funcion `plot.tclust()`
`print.DiscrFact`	A logical value indicating whether each clusters mean discriminant factor shall be plotted
`col.nodoubt`	Color of all observations not considered as to be assigned doubtfully.
`by.cluster`	Logical value indicating whether optional parameters pch and col (if present) refer to observations (FALSE) or clusters (TRUE)
`...`	Arguments to be passed to or from other methods

Details

`plot_DiscrFact_p2` displays a silhouette plot based on the discriminant factors of the observations. A solution with many large discriminant factors is not reliable. Such clusters can be identified with this silhouette plot. Thus `plot_DiscrFact_p3` displays the dataset, highlighting observations with discriminant factors greater than the given threshold. The function `plot.DiscrFact()` combines the standard plot of a `tclust` object, and the two plots introduced here.

References

García-Escudero, L.A.; Gordaliza, A.; Matrán, C. and Mayo-Iscar, A. (2011), "Exploring the number of groups in robust model-based clustering." *Statistics and Computing*, **21** pp. 585-599, <doi:10.1007/s11222-010-9194-z>

Examples

```
sig <- diag (2)
cen <- rep (1, 2)
x <- rbind(MASS::mvrnorm(360, cen * 0,   sig),
        MASS::mvrnorm(540, cen * 5,   sig * 6 - 2),
        MASS::mvrnorm(100, cen * 2.5, sig * 50))

clus.1 <- tclust(x, k = 2, alpha=0.1, restr.fact=12)
clus.2 <- tclust(x, k = 3, alpha=0.1, restr.fact=1)

dsc.1 <- DiscrFact(clus.1)
plot(dsc.1)
```

```
dsc.2 <- DiscrFact(clus.2)
plot(dsc.2)
```

`plot.rlg` *Plot an 'rlg' object*

Description

Different plots for the results of 'rlg' analysis, stored in an `rlg` object, see Details.

Usage

```
## S3 method for class 'rlg'
plot(
  x,
  which = c("all", "scores", "loadings", "eigenvalues"),
  sort = TRUE,
  ask = (which == "all" && dev.interactive(TRUE)),
  ...
)
```

Arguments

`x`	An `rlg` object to plot.
`which`	Select the required plot.
`sort`	Whether to sort.
`ask`	if `TRUE`, the user is *ask*ed before each plot, see `par(ask=.)`. Default is `ask = which=="all" && dev.interactive()`.
`...`	Other parameters to be passed to the lower level functions.

Examples

```
data (LG5data)
x <- LG5data[, 1:10]
clus <- rlg(x, d = c(2,2,2), alpha=0.1)
plot(clus, which="eigenvalues")
plot(clus, which="scores")
```

plot.tclust *Plot Method for* tclust *and* tkmeans *Objects*

Description

One and two dimensional structures are treated separately (e.g. tolerance intervals/ellipses are displayed). Higher dimensional structures are displayed by plotting the two first Fisher's canonical coordinates (evaluated by tclust::discr_coords) and derived from the final cluster assignments (trimmed observations are not taken into account). plot.tclust.Nd can be called with one or two-dimensional tclust- or tkmeans-objects too. The function fails, if store.x = FALSE is specified in the tclust() or tkmeans() call, because the original data matrix is required here.

Usage

```
## S3 method for class 'tclust'
plot(x, ...)

## S3 method for class 'tkmeans'
plot(x, ...)
```

Arguments

x	The tclust or tkmeans object to be displayed
...	Further (optional) arguments which specify the details of the resulting plot (see section "Further Arguments").

Details

The plot method for classes tclust and tkmeans.

Further Arguments

- xlab, ylab, xlim, ylim, pch, col Arguments passed to plot().
- main The title of the plot. Use "/p" for displaying the chosen parameters alpha and k or "/r" for plotting the chosen restriction.
- main.pre An optional string which is added to the plot's caption.
- sub A string specifying the subtitle of the plot. Use "/p" (default) for displaying the chosen parameters alpha and k, "/r" for plotting the chosen restriction and "/pr" for both.
- sub1 A secondary (optional) subtitle.
- labels A string specifying the type of labels to be drawn. Either labels="none" (default), labels="cluster" or labels="observation" can be specified. If specified, parameter pch is ignored.
- text A vector of length n (the number of observations) containing strings which are used as labels for each observation. If specified, the parameters labels and pch are ignored.

- `by.cluster` Logical value indicating whether parameters `pch` and `col` refer to observations (FALSE) or clusters (TRUE).
- `jitter.y` Logical value, specifying whether the drawn values shall be jittered in y-direction for better visibility of structures in 1 dimensional data.
- `tol` The tolerance interval. 95% tolerance ellipsoids (assuming normality) are plotted by default.
- `tol.col`, `tol.lty`, `tol.lwd` Vectors of length k or 1 containing the `col`, `lty` and `lwd` arguments for the tolerance ellipses/lines.

Examples

```
#--- EXAMPLE 1------------------------------
sig <- diag (2)
cen <- rep (1, 2)
x <- rbind(MASS::mvrnorm(360, cen * 0,   sig),
        MASS::mvrnorm(540, cen * 5,   sig * 6 - 2),
        MASS::mvrnorm(100, cen * 2.5, sig * 50))
# Two groups and 10\% trimming level
a <- tclust(x, k = 2, alpha = 0.1, restr.fact = 12)
plot (a)
plot (a, labels = "observation")
plot (a, labels = "cluster")
plot (a, by.cluster = TRUE)
#--- EXAMPLE 2------------------------------
sig <- diag (2)
cen <- rep (1, 2)
x <- rbind(MASS::mvrnorm(360, cen * 0,   sig),
        MASS::mvrnorm(540, cen * 5,   sig),
        MASS::mvrnorm(100, cen * 2.5, sig))
# Two groups and 10\% trimming level
a <- tkmeans(x, k = 2, alpha = 0.1)
plot (a)
plot (a, labels = "observation")
plot (a, labels = "cluster")
plot (a, by.cluster = TRUE)
```

`plot.tclustIC` *The* `plot` *method for objects of class* `tclustIC`

Description

The `plot` method for class `tclustIC`: This function implements a series of plots, which display characteristic values of each model, computed with different values for `k` and `c` for a fixed `alpha`.

Usage

```
## S3 method for class 'tclustIC'
plot(x, whichIC, main, xlab, ylab, xlim, ylim, col, lty, ...)
```

Arguments

x	The tclustIC object to be shown
whichIC	A string indicating which information criterion will be used. See the details section for more information.
main	A character-string containing the title of the plot.
xlab, ylab, xlim, ylim	Arguments passed to plot().
col	A single value or vector of line colors passed to lines.
lty	A single value or vector of line types passed to lines.
...	Arguments to be passed to or from other methods.

References

Cerioli, A., Garcia-Escudero, L.A., Mayo-Iscar, A. and Riani M. (2017). Finding the Number of Groups in Model-Based Clustering via Constrained Likelihoods, *Journal of Computational and Graphical Statistics*, pp. 404-416, https://doi.org/10.1080/10618600.2017.1390469.

Examples

```
sig <- diag (2)
cen <- rep (1, 2)
x <- rbind(MASS::mvrnorm(108, cen * 0,   sig),
        MASS::mvrnorm(162, cen * 5,   sig * 6 - 2),
        MASS::mvrnorm(30, cen * 2.5, sig * 50))

(out <- tclustIC(x, whichIC="ALL"))

plot(out)
```

---

randIndex *Calculates Rand type Indices to compare two partitions*

Description

Calculates Rand type Indices to compare two partitions

Usage

```
randIndex(c1, c2 = NULL, noisecluster = NULL)
```

Arguments

`c1`	labels of the first partition or contingency table. A numeric vector or factor containining the class labels of the first partition or a 2-dimensional numeric matrix which contains the cross-tabulation of cluster assignments.
`c2`	labels of the second partition. A numeric vector or a factor containining the class labels of the second partition. The length of the vector `c2` must be equal to the length of the vector `c1`. The second parameter is required only if `c1` is not a 2-dimensional numeric matrix.
`noisecluster`	label or number associated to the 'noise class' or 'noise level'. Number or character label which denotes the points which do not belong to any cluster. These points are not takern into account for the computation of the Rand type indexes. The default is to consider all points.

Value

A list with Rand type indexes:

- AR Adjusted Rand index. A number between -1 and 1. The adjusted Rand index is the corrected-for-chance version of the Rand index.
- RI Rand index (unadjusted). A number between 0 and 1. Rand index computes the fraction of pairs of objects for which both classification methods agree. RI ranges from 0 (no pair classified in the same way under both clusterings) to 1 (identical clusterings).
- MI Mirkin's index. A number between 0 and 1. Mirkin's index computes the percentage of pairs of objects for which both classification methods disagree. `MI=1-RI`.
- HI Hubert index. A number between -1 and 1. HI index is equal to the fraction of pairs of objects for which both classification methods agree minus the fraction of pairs of objects for which both classification methods disagree. `HI= RI-MI`.

Examples

```
##  1. randindex with the contingency table as input.
T <- matrix(c(1, 1, 0, 1, 2, 1, 0, 0, 4), nrow=3)
(ARI <- randIndex(T))

##  2. randindex with the two vectors as input.
c <- matrix(c(1, 1, 1, 2, 2, 1, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3), ncol=2, byrow=TRUE)
## c1 = numeric vector containing the labels of the first partition
c1 <- c[,1]
## c2 = numeric vector containing the labels of the second partition
c2 <- c[,2]

(ARI <- randIndex(c1,c2))

##  3. Compare ARI for iris data (true classification against tclust classification)
library(tclust)
c1 <- iris$Species  # first partition c1 is the true partition
out <- tclust(iris[, 1:4], k=3, alpha=0, restr.fact=100)
c2 <- out$cluster   # second partition c2 is the output of tclust clustering procedure
```

```
randIndex(c1,c2)

##  4. Compare ARI for iris data (exclude unassigned units from tclust).

c1 <- iris$Species      # first partition c1 is the true partition
out <- tclust(iris[,1:4], k=3, alpha=0.1, restr.fact=100)
c2 <- out$cluster      #  second partition c2 is the output of tclust clustering procedure

## Units inside c2 which contain number 0 are referred to trimmed observations
noisecluster <- 0
randIndex(c1, c2, noisecluster=0)
```

`rlg` *Robust Linear Grouping*

Description

The function `rlg()` searches for clusters around affine subspaces of dimensions given by vector `d` (the length of that vector is the number of clusters). For instance `d=c(1,2)` means that we are clustering around a line and a plane. For robustifying the estimation, a proportion `alpha` of observations is trimmed. In particular, the trimmed k-means method is represented by the rlg method, if `d=c(0,0,..0)` (a vector of length `k` with zeroes).

Usage

```
rlg(
  x,
  d,
  alpha = 0.05,
  nstart = 500,
  niter1 = 3,
  niter2 = 20,
  nkeep = 5,
  scale = FALSE,
  parallel = FALSE,
  n.cores = -1,
  trace = FALSE
)
```

Arguments

`x`	A matrix or data.frame of dimension n x p, containing the observations (row-wise).
`d`	A numeric vector of length equal to the number of clusters to be detected. Each component of vector `d` indicates the intrinsic dimension of the affine subspace where observations on that cluster are going to be clustered. All the elements of vector `d` should be smaller than the problem dimension minus 1.

`alpha`	The proportion of observations to be trimmed.
`nstart`	The number of random initializations to be performed.
`niter1`	The number of concentration steps to be performed for the nstart initializations.
`niter2`	The maximum number of concentration steps to be performed for the nkeep solutions kept for further iteration. The concentration steps are stopped, whenever two consecutive steps lead to the same data partition.
`nkeep`	The number of iterated initializations (after niter1 concentration steps) with the best values in the target function that are kept for further iterations
`scale`	A robust centering and scaling (using the median and MAD) is done if TRUE.
`parallel`	A logical value, specifying whether the nstart initializations should be done in parallel.
`n.cores`	The number of cores to use when paralellizing, only taken into account if parallel=T.
`trace`	Defines the tracing level, which is set to 0 by default. Tracing level 1 gives additional information on the stage of the iterative process.

Details

The procedure allows to deal with robust clustering around affine subspaces with an alpha proportion of trimming level by minimizing the trimmed sums of squared orthogonal residuals. Each component of vector `d` indicates the intrinsic dimension of the affine subspace where observations on that cluster are going to be clustered. Therefore a component equal to 0 on that vector implies clustering around centres, equal to 1 around lines, equal to 2 around planes and so on. The procedure so allows simultaneous clustering and dimensionality reduction.

This iterative algorithm performs "concentration steps" to improve the current cluster assignments. For approximately obtaining the global optimum, the procedure is randomly initialized `nstart` times and `niter1` concentration steps are performed for them. The `nkeep` most “promising” iterations, i.e. the `nkeep` iterated solutions with the initial best values for the target function, are then iterated until convergence or until `niter2` concentration steps are done.

Value

Returns an object of class `rlg` which is basically a list with the following elements:

- centers - A matrix of size p x k containing the location vectors (column-wise) of each cluster.
- U - A list with k elements where each element is p x d_j matrix whose d_j columns are unitary and orthogonal vectors generating the affine subspace (after subtracting the corresponding cluster’s location parameter in centers). d_j is the intrinsic dimension of the affine subspace approximation in the j-th cluster, i.e., the elements of vector d.
- cluster - A numerical vector of size n containing the cluster assignment for each observation. Cluster names are integer numbers from 1 to k, 0 indicates trimmed observations.
- obj - The value of the objective function of the best (returned) solution.
- cluster.ini - A matrix with nstart rows and number of columns equal to the number of observations and where each row shows the final clustering assignments (0 for trimmed observations) obtained after the niter1 iteration of the nstart random initializations.

- obj.ini -A numerical vector of length nstart containing the values of the target function obtained after the niter1 iteration of the nstart random initializations.
- x - The input data set.
- dimensions - The input d vector with the intrinsic dimensions. The number of clusters is the length of that vector.
- alpha - The input trimming level.

Author(s)

Javier Crespo Guerrero, Jesús Fernández Iglesias, Luis Angel Garcia Escudero, Agustin Mayo Iscar.

References

García-Escudero, L. A., Gordaliza, A., San Martin, R., Van Aelst, S., & Zamar, R. (2009). Robust linear clustering. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 71, 301-318.

Examples

```
##--- EXAMPLE 1 ------------------------------------------
data (LG5data)
x <- LG5data[, 1:10]
clus <- rlg(x, d = c(2,2,2), alpha=0.1)
plot(x, col=clus$cluster+1)
plot(clus, which="eigenvalues")
plot(clus, which="scores")

##--- EXAMPLE 2 ------------------------------------------
 data (pine)
 clus <- rlg(pine, d = c(1,1,1), alpha=0.035)
 plot(pine, col=clus$cluster+1)
```

`simula.rlg` *Simulate contaminated data set for applying rlg*

Description

Simulate alpha*100% contaminated data set for applying rlg by generating a k=3 components with equal size and # common underlying dimension q_1=q_2=q_3=q

Usage

```
simula.rlg(q = 2, p = 10, n = 200, var = 0.01, sep.means = 0, alpha = 0.05)
```

Arguments

`q`	intrinsic dimension
`p`	dimension (`p >= 2` and `p > q`)
`n`	number of observations
`var`	The smaller 'var' the smaller the scatter around the lower dimensional space
`sep.means`	Parameter controlling the location vectors separation
`alpha`	contamination level

Value

a list with the following items

- x - The generated dataset
- true - The true classification

Examples

```
res <- simula.rlg(q=5, p=200, n=150, var=0.01, sep.means=0.00)
plot(res$x,col=res$true+1)
```

`simula.tclust` *Simulate contaminated data set for applying TCLUST*

Description

Simulate 10% contaminated data set for applying TCLUST

Usage

```
simula.tclust(n, p = 4, k = 3, type = 2, balanced = 1)
```

Arguments

`n`	number of observations
`p`	dimension (p>=2 and p>q)
`k`	number of clusters (only k=3 and k=6 are allowed!!!)
`type`	1 (spherical for rest.fact=1) or 2 (elliptical for rest.fact=9^2)
`balanced`	1 (all clusters equal size) or 2 [proportions (25,30,35)% if k=3 and (12.5,15,17.5,12.5,15,17.5)% if k=6]

Value

a list with the following items

- x - The generated dataset
- true - The true classification

Examples

```
res <- simula.tclust(n=400,k=3,p=8,type=2,balanced=1)
plot(res$x,col=res$true+1)
```

`summary.DiscrFact` *The* `summary` *method for objects of class* `DiscrFact`

Description

The summary method for class `DiscrFact`.

Usage

```
## S3 method for class 'DiscrFact'
summary(object, hide.emtpy = TRUE, show.clust, show.alt, ...)
```

Arguments

`object`	An object of class `DiscrFact` as returned from `DiscrFact()`.
`hide.emtpy`	A logical value specifying whether clusters without doubtful assignment shall be hidden.
`show.clust`	A logical value specifying whether the number of doubtful assignments per cluster shall be displayed.
`show.alt`	A logical value specifying whether the alternative cluster assignment shall be displayed.
`...`	Arguments passed to or from other methods.

References

García-Escudero, L.A.; Gordaliza, A.; Matrán, C. and Mayo-Iscar, A. (2011), "Exploring the number of groups in robust model-based clustering." *Statistics and Computing*, **21** pp. 585-599, <doi:10.1007/s11222-010-9194-z>

Examples

```
 sig <- diag (2)
 cen <- rep (1, 2)
 x <- rbind(MASS::mvrnorm(360, cen * 0,   sig),
        MASS::mvrnorm(540, cen * 5,   sig * 6 - 2),
        MASS::mvrnorm(100, cen * 2.5, sig * 50)
 )

 clus.1 <- tclust(x, k = 2, alpha=0.1, restr.fact=12)
 clus.2 <- tclust(x, k = 3, alpha=0.1, restr.fact=1)

 dsc.1 <- DiscrFact(clus.1)
```

```
summary(dsc.1)

dsc.2 <- DiscrFact(clus.2)
summary(dsc.2)
```

`swissbank` *Swiss banknotes data*

Description

Six variables measured on 100 genuine and 100 counterfeit old Swiss 1000-franc bank notes (Flury and Riedwyl, 1988).

Usage

```
data(swissbank)
```

Format

A data frame containing 200 observations in 6 variables. The variables are as follows:

- `Length` Length of the bank note
- `Ht_Left` Height of the bank note, measured on the left
- `Ht_Right` Height of the bank note, measured on the right
- `IF_Lower` Distance of inner frame to the lower border
- `IF_Upper` Distance of inner frame to the upper border
- `Diagonal` Length of the diagonal

Details

Observations 1–100 are the genuine bank notes and the other 100 observations are the counterfeit bank notes.

Source

Flury, B. and Riedwyl, H. (1988). *Multivariate Statistics, A Practical Approach*, Cambridge University Press.

tclust *TCLUST method for robust clustering*

Description

This function searches for `k` (or less) clusters with different covariance structures in a data matrix `x`. Relative cluster scatter can be restricted when `restr="eigen"` by constraining the ratio between the largest and the smallest of the scatter matrices eigenvalues by a constant value `restr.fact`. Relative cluster scatters can be also restricted with `restr="deter"` by constraining the ratio between the largest and the smallest of the scatter matrices' determinants.

For robustifying the estimation, a proportion `alpha` of observations is trimmed. In particular, the trimmed k-means method is represented by the `tclust()` method, by setting parameters `restr.fact=1`, `opt="HARD"` and `equal.weights=TRUE`.

Usage

```
tclust(
  x,
  k,
  alpha = 0.05,
  nstart = 500,
  niter1 = 3,
  niter2 = 20,
  nkeep = 5,
  iter.max,
  equal.weights = FALSE,
  restr = c("eigen", "deter"),
  restr.fact = 12,
  cshape = 1e+10,
  opt = c("HARD", "MIXT"),
  center = FALSE,
  scale = FALSE,
  store_x = TRUE,
  parallel = FALSE,
  n.cores = -1,
  zero_tol = 1e-16,
  drop.empty.clust = TRUE,
  trace = 0
)
```

Arguments

`x`	A matrix or data.frame of dimension n x p, containing the observations (row-wise).
`k`	The number of clusters initially searched for.
`alpha`	The proportion of observations to be trimmed.

`nstart`	The number of random initializations to be performed.
`niter1`	The number of concentration steps to be performed for the nstart initializations.
`niter2`	The maximum number of concentration steps to be performed for the `nkeep` solutions kept for further iteration. The concentration steps are stopped, whenever two consecutive steps lead to the same data partition.
`nkeep`	The number of iterated initializations (after niter1 concentration steps) with the best values in the target function that are kept for further iterations
`iter.max`	(deprecated, use the combination `nkeep, niter1` and `niter2`) The maximum number of concentration steps to be performed. The concentration steps are stopped, whenever two consecutive steps lead to the same data partition.
`equal.weights`	A logical value, specifying whether equal cluster weights shall be considered in the concentration and assignment steps.
`restr`	Restriction type to control relative cluster scatters. The default value is `restr="eigen"`, so that the maximal ratio between the largest and the smallest of the scatter matrices eigenvalues is constrained to be smaller then or equal to `restr.fact` (Garcia-Escudero, Gordaliza, Matran, and Mayo-Iscar, 2008). Alternatively, `restr="deter"` imposes that the maximal ratio between the largest and the smallest of the scatter matrices determinants is smaller or equal than `restr.fact` (see Garcia-Escudero, Mayo-Iscar and Riani, 2020)
`restr.fact`	The constant `restr.fact >= 1` constrains the allowed differences among group scatters in terms of eigenvalues ratio (if `restr="eigen"`) or determinant ratios (if `restr="deter"`). Larger values imply larger differences of group scatters, a value of 1 specifies the strongest restriction.
`cshape`	constraint to apply to the shape matrices, `cshape >= 1`, (see Garcia-Escudero, Mayo-Iscar and Riani, 2020)). This options only works if `restr=='deter'`. In this case the default value is `cshape=1e10` to ensure the procedure is (virtually) affine equivariant. On the other hand, `cshape` values close to 1 would force the clusters to be almost spherical (without necessarily the same scatters if `restr.fact` is strictly greater than 1).
`opt`	Define the target function to be optimized. A classification likelihood target function is considered if `opt="HARD"` and a mixture classification likelihood if `opt="MIXT"`.
`center`	Optional centering of the data: a function or a vector of length p which can optionally be specified for centering x before calculation
`scale`	Optional scaling of the data: a function or a vector of length p which can optionally be specified for scaling x before calculation
`store_x`	A logical value, specifying whether the data matrix `x` shall be included in the result object. By default this value is set to `TRUE`, because some of the plotting functions depend on this information. However, when big data matrices are handled, the result object's size can be decreased noticeably when setting this parameter to `FALSE`.
`parallel`	A logical value, specifying whether the nstart initializations should be done in parallel.
`n.cores`	The number of cores to use when paralellizing, only taken into account if `parallel=TRUE`.

`zero_tol`	The zero tolerance used. By default set to 1e-16.
`drop.empty.clust`	Logical value specifying, whether empty clusters shall be omitted in the resulting object. (The result structure does not contain center and covariance estimates of empty clusters anymore. Cluster names are reassigned such that the first l clusters (l <= k) always have at least one observation.
`trace`	Defines the tracing level, which is set to 0 by default. Tracing level 1 gives additional information on the stage of the iterative process.

Details

The procedure allows to deal with robust clustering with an `alpha` proportion of trimming level and searching for `k` clusters. We are considering classification trimmed likelihood when using `opt="HARD"` so that "hard" or "crisp" clustering assignments are done. On the other hand, mixture trimmed likelihood are applied when using `opt="MIXT"` so providing a kind of clusters "posterior" probabilities for the observations. Relative cluster scatter can be restricted when `restr="eigen"` by constraining the ratio between the largest and the smallest of the scatter matrices eigenvalues by a constant value `restr.fact`. Setting `restr.fact=1`, yields the strongest restriction, forcing all clusters to be spherical and equally scattered. Relative cluster scatters can be also restricted with `restr="deter"` by constraining the ratio between the largest and the smallest of the scatter matrices' determinants.

This iterative algorithm performs "concentration steps" to improve the current cluster assignments. For approximately obtaining the global optimum, the procedure is randomly initialized `nstart` times and `niter1` concentration steps are performed for them. The `nkeep` most "promising" iterations, i.e. the `nkeep` iterated solutions with the initial best values for the target function, are then iterated until convergence or until `niter2` concentration steps are done.

The parameter `restr.fact` defines the cluster scatter matrices restrictions, which are applied on all clusters during each concentration step. It restricts the ratio between the maximum and minimum eigenvalue of all clusters' covariance structures to that parameter. Setting `restr.fact=1`, yields the strongest restriction, forcing all clusters to be spherical and equally scattered.

Cluster components with similar sizes are favoured when considering `equal.weights=TRUE` while `equal.weights=FALSE` admits possible different prior probabilities for the components and it can easily return empty clusters when the number of clusters is greater than apparently needed.

Value

The function returns the following values:

- cluster - A numerical vector of size `n` containing the cluster assignment for each observation. Cluster names are integer numbers from 1 to k, 0 indicates trimmed observations. Note that it could be empty clusters with no observations when `equal.weights=FALSE`.
- obj - The value of the objective function of the best (returned) solution.
- NlogL - A value related to the classification log-likelihood of the best (returned) solution. If `opt=="HARD"`, `NlogL = -2*obj`.
- size - An integer vector of size k, returning the number of observations contained by each cluster.
- weights - Vector of Cluster weights

- centers - A matrix of size p x k containing the centers (column-wise) of each cluster.
- cov - An array of size p x p x k containing the covariance matrices of each cluster.
- code - A numerical value indicating if the concentration steps have converged for the returned solution (2).
- posterior - A matrix with k columns that contains the posterior probabilities of membership of each observation (row-wise) to the k clusters. This posterior probabilities are 0-1 values in the `opt="HARD"` case. Trimmed observations have 0 membership probabilities to all clusters.
- MIXMIX - BIC which based on the parameters estimated through the mixture log-likelihood and the maximized mixture likelihood as goodness of fit measure. This output is present only if `opt="MIXT"`.
- MIXMIX - BIC which uses the classification likelihood based on parameters estimated through the mixture likelihood (In some books this quantity is called ICL). This output is present only if `opt="MIXT"`.
- CLACLA - BIC which uses the classification likelihood based on parameters estimated using the classification likelihood. This output is present only if `opt="HARD"`.
- cluster.ini - A matrix with nstart rows and number of columns equal to the number of observations and where each row shows the final clustering assignments (0 for trimmed observations) obtained after the `niter1` iteration of the `nstart` random initializations.
- obj.ini - A numerical vector of length `nstart` containing the values of the target function obtained after the `niter1` iteration of the `nstart` random initializations.
- x - The input data set.
- k - The input number of clusters.
- alpha - The input trimming level.

Author(s)

Javier Crespo Guerrero, Luis Angel Garcia Escudero, Agustin Mayo Iscar.

References

Fritz, H.; Garcia-Escudero, L.A.; Mayo-Iscar, A. (2012), "tclust: An R Package for a Trimming Approach to Cluster Analysis". Journal of Statistical Software, 47(12), 1-26. URL http://www.jstatsoft.org/v47/i12/

Garcia-Escudero, L.A.; Gordaliza, A.; Matran, C. and Mayo-Iscar, A. (2008), "A General Trimming Approach to Robust Cluster Analysis". Annals of Statistics, Vol.36, 1324–1345.

García-Escudero, L. A., Gordaliza, A. and Mayo-Íscar, A. (2014). A constrained robust proposal for mixture modeling avoiding spurious solutions. Advances in Data Analysis and Classification, 27–43.

García-Escudero, L. A., and Mayo-Íscar, A. and Riani, M. (2020). Model-based clustering with determinant-and-shape constraint. Statistics and Computing, 30, 1363–1380.]

Examples

```
##--- EXAMPLE 1 ------------------------------------------
sig <- diag(2)
```

```
cen <- rep(1,2)
x <- rbind(MASS::mvrnorm(360, cen * 0,   sig),
           MASS::mvrnorm(540, cen * 5,   sig * 6 - 2),
           MASS::mvrnorm(100, cen * 2.5, sig * 50))

## Two groups and 10\% trimming level
clus <- tclust(x, k = 2, alpha = 0.1, restr.fact = 8)

plot(clus)
plot(clus, labels = "observation")
plot(clus, labels = "cluster")

## Three groups (one of them very scattered) and 0\% trimming level
clus <- tclust(x, k = 3, alpha=0.0, restr.fact = 100)

plot(clus)

##--- EXAMPLE 2 ------------------------------------------
data(geyser2)
(clus <- tclust(geyser2, k = 3, alpha = 0.03))

plot(clus)



##--- EXAMPLE 3 ------------------------------------------
data(M5data)
x <- M5data[, 1:2]

clus.a <- tclust(x, k = 3, alpha = 0.1, restr.fact =  1,
                  restr = "eigen", equal.weights = TRUE)
clus.b <- tclust(x, k = 3, alpha = 0.1, restr.fact =  50,
                   restr = "eigen", equal.weights = FALSE)
clus.c <- tclust(x, k = 3, alpha = 0.1, restr.fact =  1,
                  restr = "deter", equal.weights = TRUE)
clus.d <- tclust(x, k = 3, alpha = 0.1, restr.fact = 50,
                  restr = "deter", equal.weights = FALSE)

pa <- par(mfrow = c (2, 2))
plot(clus.a, main = "(a)")
plot(clus.b, main = "(b)")
plot(clus.c, main = "(c)")
plot(clus.d, main = "(d)")
par(pa)

##--- EXAMPLE 4 ------------------------------------------

data (swissbank)
## Two clusters and 8\% trimming level
(clus <- tclust(swissbank, k = 2, alpha = 0.08, restr.fact = 50))

## Pairs plot of the clustering solution
pairs(swissbank, col = clus$cluster + 1)
```

```
## Two coordinates
plot(swissbank[, 4], swissbank[, 6], col = clus$cluster + 1,
     xlab = "Distance of the inner frame to lower border",
     ylab = "Length of the diagonal")
plot(clus)

## Three clusters and 0\% trimming level
clus<- tclust(swissbank, k = 3, alpha = 0.0, restr.fact = 110)

## Pairs plot of the clustering solution
pairs(swissbank, col = clus$cluster + 1)

## Two coordinates
plot(swissbank[, 4], swissbank[, 6], col = clus$cluster + 1,
      xlab = "Distance of the inner frame to lower border",
      ylab = "Length of the diagonal")

plot(clus)

##--- EXAMPLE 5 ------------------------------------------
 data(M5data)
 x <- M5data[, 1:2]

 ## Classification trimmed likelihood approach
 clus.a <- tclust(x, k = 3, alpha = 0.1, restr.fact =  50,
                   opt="HARD", restr = "eigen", equal.weights = FALSE)
## Mixture trimmed likelihood approach
 clus.b <- tclust(x, k = 3, alpha = 0.1, restr.fact =  50,
                   opt="MIXT", restr = "eigen", equal.weights = FALSE)

## Hard 0-1 cluster assignment (all 0 if trimmed unit)
head(clus.a$posterior)

## Posterior probabilities cluster assignment for the
##  mixture approach (all 0 if trimmed unit)
head(clus.b$posterior)
```

tclustIC

Performs cluster analysis by calling `tclust` *for different number of groups* `k` *and restriction factors* `c`

Description

Computes the values of BIC (MIXMIX), ICL (MIXCLA) or CLA (CLACLA), for different values of `k` (number of groups) and different values of `c` (restriction factor), for a prespecified level of trimming (the last two letters in the name stand for 'Information Criterion').

Usage

```
tclustIC(
  x,
  kk = 1:5,
  cc = c(1, 2, 4, 8, 16, 32, 64, 128),
  alpha = 0.05,
  whichIC = c("ALL", "MIXMIX", "MIXCLA", "CLACLA"),
  parallel = FALSE,
  n.cores = -1,
  trace = FALSE,
  ...
)
```

Arguments

`x`	A matrix or data frame of dimension n x p, containing the observations (row-wise).
`kk`	an integer vector specifying the number of mixture components (clusters) for which the information criteria are be calculated. By default `kk=1:5`.
`cc`	an vector specifying the values of the restriction factor which have to be considered. By default `cc=c(1, 2, 4, 8, 16, 32, 64, 128)`.
`alpha`	The proportion of observations to be trimmed.
`whichIC`	A character value which specifies which information criteria must be computed for each `k` (number of groups) and each value of the restriction factor `c`. Possible values for `whichIC` are: • "MIXMIX": a mixture model is fitted and for computing the information criterion the mixture likelihood is used. This option corresponds to the use of the Bayesian Information criterion (BIC). In output just the matrix `MIXMIX` is given. • "MIXCLA": a mixture model is fitted but to compute the information criterion the classification likelihood is used. This option corresponds to the use of the Integrated Complete Likelihood (ICL). In the output just the matrix `MIXCLA` is given. • "CLACLA": everything is based on the classification likelihood. This information criterion will be called CLA. In the output just the matrix `CLACLA` is given. • "ALL": both classification and mixture likelihood are used. In this case all three information criteria CLA, ICL and BIC are computed. In the output all three matrices `MIXMIX`, `MIXCLA` and `CLACLA` are given.
`parallel`	A logical value, specifying whether the calls to `tclust` should be done in parallel.
`n.cores`	The number of cores to use when paralellizing, only taken into account if `parallel=TRUE`.
`trace`	Whether to print intermediate results. Default is `trace=FALSE`.
`...`	Further arguments (as e.g. `restr`), passed to `tclust`

Value

The functions `print()` and `summary()` are used to obtain and print a summary of the results. The function returns an S3 object of type `tclustIC` containing the following components:

- call the matched call
- kk a vector containing the values of `k` (number of components) which have been considered. This vector is identical to the optional argument `kk` (default is `kk=1:5`.
- cc a vector containing the values of `c` (values of the restriction factor) which have been considered. This vector is identical to the optional argument `cc` (defalt is `cc=c(1, 2, 4, 8, 16, 32, 64, 128)`.
- alpha trimming level
- whichIC Information criteria used
- CLACLA a matrix of size `length(kk)-times-length(cc)` containinig the value of the penalized classification likelihood. This output is present only if `whichIC="CLACLA"` or `whichIC="ALL"`.
- IDXCLA a matrix of lists of size `length(kk)-times-length(cc)` containinig the assignment of each unit using the classification model. This output is present only if `whichIC="CLACLA"` or `whichIC="ALL"`.
- MIXMIX a matrix of size `length(kk)-times-length(cc)` containinig the value of the penalized mixtrue likelihood. This output is present only if `whichIC="MIXMIX"` or `whichIC="ALL"`.
- IDXMIX a matrix of lists of size `length(kk)-times-length(cc)` containinig the assignment of each unit using the classification model. This output is present only if `whichIC="MIXMIX"` or `whichIC="ALL"`.
- MIXCLA a matrix of size `length(kk)-times-length(cc)` containinig the value of the ICL criterion. This output is present only if `whichIC="MIXCLA"` or `whichIC="ALL"`.

References

Cerioli, A., Garcia-Escudero, L.A., Mayo-Iscar, A. and Riani M. (2017). Finding the Number of Groups in Model-Based Clustering via Constrained Likelihoods, *Journal of Computational and Graphical Statistics*, pp. 404-416, https://doi.org/10.1080/10618600.2017.1390469.

See Also

`tclust`

Examples

```
#--- EXAMPLE 1 ------------------------------------------

data(geyser2)
(out <- tclustIC(geyser2, whichIC="MIXMIX", alpha=0.1))
summary(out)
## Find the smallest value inside the table and write the corresponding
## values of k (number of groups) and c (restriction factor)
inds <- which(out$MIXMIX == min(out$MIXMIX), arr.ind=TRUE)
vals <- out$MIXMIX[inds]
cat("\nThe smallest value of the IC is ", vals,
    " and takes place for k=", out$kk[inds[1]], " and c=",
    out$cc[inds[2]], "\n")


#--- EXAMPLE 2 ------------------------------------------

data(flea)
Y <- as.matrix(flea[, 1:(ncol(flea)-1)])    # select only the numeric variables
rownames(Y) <- 1:nrow(Y)
head(Y)

(out <- tclustIC(Y, whichIC="CLACLA", alpha=0.1))
summary(out)
## Find the smallest value inside the table and write the corresponding
## values of k (number of groups) and c (restriction factor)
inds <- which(out$CLACLA == min(out$CLACLA), arr.ind=TRUE)
vals <- out$CLACLA[inds]
cat("\nThe Smallest value of the IC is ", vals,
    " and takes place for k=", out$kk[inds[1]], " and c=",
    out$cc[inds[2]], "\n")


#--- EXAMPLE 3 ------------------------------------------

data(swissbank)
(out <- tclustIC(swissbank, whichIC="ALL"))

plot(out)  ##  --> selecting k=3, c=128

##  the selected model
plot(tclust(swissbank, k = 3, alpha = 0.1, restr.fact = 128))
```

tkmeans — *TKMEANS method for robust K-means clustering*

Description

This function searches for `k` (or less) spherical clusters in a data matrix `x`, whereas the `ceiling(alpha n)` most outlying observations are trimmed.

Usage

```
tkmeans(
  x,
  k,
  alpha = 0.05,
  nstart = 500,
  niter1 = 3,
  niter2 = 20,
  nkeep = 5,
  iter.max,
  points = NULL,
  center = FALSE,
  scale = FALSE,
  store_x = TRUE,
  parallel = FALSE,
  n.cores = -1,
  zero_tol = 1e-16,
  drop.empty.clust = TRUE,
  trace = 0
)
```

Arguments

`x`	A matrix or data.frame of dimension n x p, containing the observations (row-wise).
`k`	The number of clusters initially searched for.
`alpha`	The proportion of observations to be trimmed.
`nstart`	The number of random initializations to be performed.
`niter1`	The number of concentration steps to be performed for the nstart initializations.
`niter2`	The maximum number of concentration steps to be performed for the `nkeep` solutions kept for further iteration. The concentration steps are stopped, whenever two consecutive steps lead to the same data partition.
`nkeep`	The number of iterated initializations (after niter1 concentration steps) with the best values in the target function that are kept for further iterations
`iter.max`	(deprecated, use the combination `nkeep`, `niter1` and `niter2`) The maximum number of concentration steps to be performed. The concentration steps are stopped, whenever two consecutive steps lead to the same data partition.
`points`	Optional initial mean vectors, `NULL` or a matrix with `k` vectors used as means to initialize the algorithm. If initial mean vectors are specified, `nstart` should be 1 (otherwise the same initial means are used for all runs).
`center`	Optional centering of the data: a function or a vector of length p which can optionally be specified for centering x before calculation
`scale`	Optional scaling of the data: a function or a vector of length p which can optionally be specified for scaling x before calculation
`store_x`	A logical value, specifying whether the data matrix `x` shall be included in the result object. By default this value is set to `TRUE`, because some of the plotting functions depend on this information. However, when big data matrices are handled, the result object's size can be decreased noticeably when setting this parameter to `FALSE`.
`parallel`	A logical value, specifying whether the nstart initializations should be done in parallel.

`n.cores`	The number of cores to use when paralellizing, only taken into account if parallel=TRUE.
`zero_tol`	The zero tolerance used. By default set to 1e-16.
`drop.empty.clust`	Logical value specifying, whether empty clusters shall be omitted in the resulting object. (The result structure does not contain center estimates of empty clusters anymore. Cluster names are reassigned such that the first l clusters (l <= k) always have at least one observation.
`trace`	Defines the tracing level, which is set to 0 by default. Tracing level 1 gives additional information on the stage of the iterative process.

Value

The function returns the following values:

- cluster - A numerical vector of size `n` containing the cluster assignment for each observation. Cluster names are integer numbers from 1 to k, 0 indicates trimmed observations. Note that it could be empty clusters with no observations when `equal.weights=FALSE`.
- obj - The value of the objective function of the best (returned) solution.
- size - An integer vector of size k, returning the number of observations contained by each cluster.
- centers - A matrix of size p x k containing the centers (column-wise) of each cluster.
- code - A numerical value indicating if the concentration steps have converged for the returned solution (2).
- cluster.ini - A matrix with nstart rows and number of columns equal to the number of observations and where each row shows the final clustering assignments (0 for trimmed observations) obtained after the `niter1` iteration of the `nstart` random initializations.
- obj.ini - A numerical vector of length `nstart` containing the values of the target function obtained after the `niter1` iteration of the `nstart` random initializations.
- x - The input data set.
- k - The input number of clusters.
- alpha - The input trimming level.

Author(s)

Valentin Todorov, Luis Angel Garcia Escudero, Agustin Mayo Iscar.

References

Cuesta-Albertos, J. A.; Gordaliza, A. and Matrán, C. (1997), "Trimmed k-means: an attempt to robustify quantizers". Annals of Statistics, Vol. 25 (2), 553-576.

Examples

```
##--- EXAMPLE 1 ------------------------------------------
sig <- diag(2)
cen <- rep(1,2)
x <- rbind(MASS::mvrnorm(360, cen * 0,   sig),
           MASS::mvrnorm(540, cen * 5,   sig),
           MASS::mvrnorm(100, cen * 2.5, sig))

## Two groups and 10\% trimming level
(clus <- tkmeans(x, k = 2, alpha = 0.1))

plot(clus)
plot(clus, labels = "observation")
plot(clus, labels = "cluster")

#--- EXAMPLE 2 ------------------------------------------
data(geyser2)
(clus <- tkmeans(geyser2, k = 3, alpha = 0.03))
plot(clus)
```

`wholesale` *Wholesale customers dataset*

Description

The data set refers to clients of a wholesale distributor. It includes the annual spending in monetary units on diverse product categories.

Usage

```
data(wholesale)
```

Format

A data frame containing 440 observations in 8 variables (6 numerical and two categorical). The variables are as follows:

- `Region` Customers' Region - Lisbon (coded as 1), Porto (coded as 2) or Other (coded as 3)
- `Fresh` Annual spending on fresh products
- `Milk` Annual spending on milk products
- `Grocery` Annual spending on grocery products
- `Frozen` Annual spending on frozen products
- `Detergents` Annual spending on detergents and paper products
- `Delicatessen` Annual spending on and delicatessen products
- `Channel` Customers' Channel - Horeca (Hotel/Restaurant/Café) or Retail channel. Horeca is coded as 1 and Retail channel is coded as 2

Source

Abreu, N. (2011). Analise do perfil do cliente Recheio e desenvolvimento de um sistema promocional. Mestrado em Marketing, ISCTE-IUL, Lisbon. url=https://api.semanticscholar.org/CorpusID:124027622

Examples

```
#--- EXAMPLE 1 ------------------------------------------
data (wholesale)
x <- wholesale[, -c(1, ncol(wholesale))]
clus <- tclust(x, k=3, alpha=0.1, nstart=200, niter1=3, niter2=17,
   nkeep=10, opt="HARD", equal.weights=FALSE, restr.fact=50, trace=TRUE)
 plot (x, col=clus$cluster+1)
 plot(clus)
```

Index