Package 'T4cluster'

October 12, 2022

Type Package

Title Tools for Cluster Analysis

Version 0.1.2

Encoding UTF-8

Description Cluster analysis is one of the most fundamental problems in data science. We provide a variety of algorithms from clustering to the learning on the space of partitions. See Hennig, Meila, and Rocci (2016, ISBN:9781466551886) for general exposition to cluster analysis.

License MIT + file LICENSE

Imports Rcpp (>= 1.0.5), Rdpack, Rdimtools, ADMM, MASS, fda, ggplot2, lpSolve, maotai, mclustcomp, rstiefel, scatterplot3d, stats, utils

URL `https://kisungyou.com/T4cluster/`

BugReports `https://github.com/kisungyou/T4cluster/issues`

LinkingTo Rcpp, RcppArmadillo

RdMacros Rdpack

RoxygenNote 7.1.1

Depends R (>= 2.10)

NeedsCompilation yes

Author Kisung You [aut, cre] (<`https://orcid.org/0000-0002-8584-459X`>)

Maintainer Kisung You <`kisungyou@outlook.com`>

Repository CRAN

Date/Publication 2021-08-16 07:20:06 UTC

R topics documented:

compare.adjrand 2
compare.rand 4
dpmeans 5
EKSS 6

funhclust 8
funkmeans03A 9
gen3S 11
genDONUTS 12
genLP 13
genSMILEY 14
gmm 15
gmm03F 16
gmm11R 18
gmm16G 19
gskmeans 21
household 22
kmeans 23
kmeans18B 24
kmeanspp 25
LRR 27
LRSC 28
LSR 30
MSM 31
pcm 33
predict.MSM 34
psm 35
sc05Z 36
sc09G 37
sc10Z 39
sc11Y 40
sc12L 42
scNJW 43
scSM 45
scUL 46
spkmeans 48
SSC 49
SSQP 51

Index **53**

`compare.adjrand` *(+) Adjusted Rand Index*

Description

Compute Adjusted Rand index between two clusterings. Please note that the value can yield negative value.

Usage

```
compare.adjrand(x, y)
```

Arguments

`x`	1st cluster label vector of length-n.
`y`	2nd cluster label vector of length-n.

Value

Adjusted Rand Index value.

See Also

`compare.rand`

Examples

```
# -------------------------------------------------------------
#         true label vs. clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## CLUSTERING WITH DIFFERENT K VALUES
vec_k  = 2:7
vec_cl = list()
for (i in 1:length(vec_k)){
  vec_cl[[i]] = T4cluster::kmeans(X, k=round(vec_k[i]))$cluster
}

## COMPUTE COMPARISON INDICES
vec_comp = rep(0, length(vec_k))
for (i in 1:length(vec_k)){
  vec_comp[i] = compare.adjrand(vec_cl[[i]], lab)
}

## VISUALIZE
opar <- par(no.readonly=TRUE)
plot(vec_k, vec_comp, type="b", lty=2, xlab="number of clusters",
     ylab="comparison index", main="Adjusted Rand Index with true k=3")
abline(v=3, lwd=2, col="red")
par(opar)
```

compare.rand *(+) Rand Index*

Description

Compute Rand index between two clusterings. It has a value between 0 and 1 where 0 indicates two clusterings do not agree and 1 exactly the same.

Usage

```
compare.rand(x, y)
```

Arguments

x	1st cluster label vector of length-n.
y	2nd cluster label vector of length-n.

Value

Rand Index value.

References

Rand WM (1971). "Objective Criteria for the Evaluation of Clustering Methods." *Journal of the American Statistical Association*, **66**(336), 846. ISSN 01621459.

Examples

```
# -------------------------------------------------------------
#         true label vs. clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## CLUSTERING WITH DIFFERENT K VALUES
vec_k  = 2:7
vec_cl = list()
for (i in 1:length(vec_k)){
  vec_cl[[i]] = T4cluster::kmeans(X, k=round(vec_k[i]))$cluster
}

## COMPUTE COMPARISON INDICES
vec_comp = rep(0, length(vec_k))
for (i in 1:length(vec_k)){
  vec_comp[i] = compare.rand(vec_cl[[i]], lab)
}
```

```
## VISUALIZE
opar <- par(no.readonly=TRUE)
plot(vec_k, vec_comp, type="b", lty=2, xlab="number of clusters",
     ylab="comparison index", main="Rand Index with true k=3")
abline(v=3, lwd=2, col="red")
par(opar)
```

---

dpmeans *DP-Means Clustering*

Description

DP-means is a non-parametric clustering method motivated by DP mixture model in that the number of clusters is determined by a parameter λ. The larger the λ value is, the smaller the number of clusters is attained. In addition to the original paper, we added an option to randomly permute an order of updating for each observation's membership as a common heuristic in the literature of cluster analysis.

Usage

```
dpmeans(data, lambda = 0.1, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`lambda`	a threshold to define a new cluster (default: 0.1).
`...`	extra parameters including **maxiter** the maximum number of iterations (default: 10). **eps** the stopping criterion for iterations (default: 1e-5). **permute** a logical; `TRUE` if random order for update is used, `FALSE` otherwise (default).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1 : k$).

algorithm name of the algorithm.

References

Kulis B, Jordan MI (2012). "Revisiting K-Means: New Algorithms via Bayesian Nonparametrics." In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, 1131–1138. ISBN 978-1-4503-1285-1.

Examples

```
# -------------------------------------------------------------
#            clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT LAMBDA VALUES
dpm1 = dpmeans(X, lambda=1)$cluster
dpm2 = dpmeans(X, lambda=5)$cluster
dpm3 = dpmeans(X, lambda=25)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=dpm1, pch=19, main="dpmeans: lambda=1")
plot(X2d, col=dpm2, pch=19, main="dpmeans: lambda=5")
plot(X2d, col=dpm3, pch=19, main="dpmeans: lambda=25")
par(opar)
```

EKSS *Ensembles of K-Subspaces*

Description

Ensembles of K-Subspaces method exploits multiple runs of K-Subspace Clustering and uses consensus framework to aggregate multiple clustering results to mitigate the effect of random initializations. When the results are merged, it zeros out $n - q$ number of values in a co-occurrence matrix. The paper suggests to use large number of runs (`B`) where each run may not require large number of iterations (`iter`) since the main assumption of the algorithm is to utilize multiple partially-correct information. At the extreme case, iteration `iter` may be set to 0 for which the paper denotes it as EKSS-0.

Usage

```
EKSS(data, k = 2, d = 2, q = floor(nrow(data) * 0.75), B = 500, iter = 0)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).

`d`	candidate dimension for each subspace (default: 2).
`q`	threshold; the number of smaller values to be zeroed out (default: 0.75*n).
`B`	the number of ensembles/runs (default: 500).
`iter`	the number of iteration for each run (default: 0).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1 : k$).

algorithm name of the algorithm.

References

Lipor J, Hong D, Tan YS, Balzano L (2021). "Subspace Clustering Using Ensembles of K-Subspaces." arXiv:1709.04744.

Examples

```
## generate a toy example
set.seed(10)
tester = genLP(n=100, nl=2, np=1, iso.var=0.1)
data   = tester$data
label  = tester$class

## do PCA for data reduction
proj = base::eigen(stats::cov(data))$vectors[,1:2]
dat2 = data%*%proj

## run EKSS algorithm with k=2,3,4 with EKSS-0 and 5 iterations
out2zero = EKSS(data, k=2)
out3zero = EKSS(data, k=3)
out4zero = EKSS(data, k=4)

out2iter = EKSS(data, k=2, iter=5)
out3iter = EKSS(data, k=3, iter=5)
out4iter = EKSS(data, k=4, iter=5)

## extract label information
lab2zero = out2zero$cluster
lab3zero = out3zero$cluster
lab4zero = out4zero$cluster

lab2iter = out2iter$cluster
lab3iter = out3iter$cluster
lab4iter = out4iter$cluster

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,3))
```

```
plot(dat2, pch=19, cex=0.9, col=lab2zero, main="EKSS-0:K=2")
plot(dat2, pch=19, cex=0.9, col=lab3zero, main="EKSS-0:K=3")
plot(dat2, pch=19, cex=0.9, col=lab4zero, main="EKSS-0:K=4")
plot(dat2, pch=19, cex=0.9, col=lab2iter, main="EKSS iter:K=2")
plot(dat2, pch=19, cex=0.9, col=lab3iter, main="EKSS iter:K=3")
plot(dat2, pch=19, cex=0.9, col=lab4iter, main="EKSS iter:K=4")
par(opar)
```

funhclust *Functional Hierarchical Clustering*

Description

Given N curves $\gamma_1(t), \gamma_2(t), \ldots, \gamma_N(t) : I \rightarrow \mathbf{R}$, perform hierarchical agglomerative clustering with **fastcluster** package's implementation of the algorithm. Dissimilarity for curves is measured by L_p metric.

Usage

```
funhclust(
  fdobj,
  p = 2,
  method = c("single", "complete", "average", "mcquitty", "ward.D", "ward.D2",
    "centroid", "median"),
  members = NULL
)
```

Arguments

`fdobj`	a `'fd'` functional data object of N curves by the **fda** package.
`p`	an exponent in L_p formalism (default: 2).
`method`	agglomeration method to be used. This must be one of `"single"`, `"complete"`, `"average"`, `"mcquitty"`, `"ward.D"`, `"ward.D2"`, `"centroid"` or `"median"`.
`members`	`NULL` or a vector whose length equals the number of observations. See `hclust` for details.

Value

an object of class `hclust`. See `hclust` for details.

References

Ferreira L, Hitchcock DB (2009). "A Comparison of Hierarchical Methods for Clustering Functional Data." *Communications in Statistics - Simulation and Computation*, **38**(9), 1925–1949. ISSN 0361-0918, 1532-4141.

Examples

```
# -------------------------------------------------------------
#                    two types of curves
#
# type 1 : sin(x) + perturbation; 20 OF THESE ON [0, 2*PI]
# type 2 : cos(x) + perturbation; 20 OF THESE ON [0, 2*PI]
# -------------------------------------------------------------
## PREPARE : USE 'fda' PACKAGE
#  Generate Raw Data
datx = seq(from=0, to=2*pi, length.out=100)
daty = array(0,c(100, 40))
for (i in 1:20){
  daty[,i]    = sin(datx) + rnorm(100, sd=0.1)
  daty[,i+20] = cos(datx) + rnorm(100, sd=0.1)
}
#  Wrap as 'fd' object
mybasis <- fda::create.bspline.basis(c(0,2*pi), nbasis=10)
myfdobj <- fda::smooth.basis(datx, daty, mybasis)$fd

## RUN THE ALGORITHM
hcsingle = funhclust(myfdobj, method="single")

## VISUALIZE
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
matplot(datx, daty[,1:20],  type="l", main="Curves Type 1")
matplot(datx, daty[,21:40], type="l", main="Curves Type 2")
plot(hcsingle, main="hclust with 'single' linkage")
par(opar)
```

`funkmeans03A` *Functional K-Means Clustering by Abraham et al. (2003)*

Description

Given N curves $\gamma_1(t), \gamma_2(t), \ldots, \gamma_N(t) : I \rightarrow \mathbf{R}$, perform k-means clustering on the coefficients from the functional data expanded by B-spline basis. Note that in the original paper, authors used B-splines as the choice of basis due to nice properties. However, we allow other types of basis as well for convenience.

Usage

```
funkmeans03A(fdobj, k = 2, ...)
```

Arguments

`fdobj`	a `'fd'` functional data object of N curves by the **fda** package.
`k`	the number of clusters (default: 2).
`...`	extra parameters including **maxiter** the maximum number of iterations (default: 10). **nstart** the number of random initializations (default: 5).

Value

a named list of S3 class `T4cluster` containing

cluster a length-N vector of class labels (from $1 : k$).

mean a `'fd'` object of k mean curves.

algorithm name of the algorithm.

References

Abraham C, Cornillon PA, Matzner-Lober E, Molinari N (2003). “Unsupervised Curve Clustering Using B-Splines.” *Scandinavian Journal of Statistics*, **30**(3), 581–595. ISSN 0303-6898, 1467-9469.

Examples

```
# -------------------------------------------------------------
#                    two types of curves
#
# type 1 : sin(x) + perturbation; 20 OF THESE ON [0, 2*PI]
# type 2 : cos(x) + perturbation; 20 OF THESE ON [0, 2*PI]
# type 3 : sin(x) + cos(0.5x)   ; 20 OF THESE ON [0, 2*PI]
# -------------------------------------------------------------
## PREPARE : USE 'fda' PACKAGE
#  Generate Raw Data
datx = seq(from=0, to=2*pi, length.out=100)
daty = array(0,c(100, 60))
for (i in 1:20){
  daty[,i]    = sin(datx) + rnorm(100, sd=0.5)
  daty[,i+20] = cos(datx) + rnorm(100, sd=0.5)
  daty[,i+40] = sin(datx) + cos(0.5*datx) + rnorm(100, sd=0.5)
}
#  Wrap as 'fd' object
mybasis <- fda::create.bspline.basis(c(0,2*pi), nbasis=10)
myfdobj <- fda::smooth.basis(datx, daty, mybasis)$fd

## RUN THE ALGORITHM WITH K=2,3,4
fk2 = funkmeans03A(myfdobj, k=2)
fk3 = funkmeans03A(myfdobj, k=3)
fk4 = funkmeans03A(myfdobj, k=4)

## FUNCTIONAL PCA FOR VISUALIZATION
embed = fda::pca.fd(myfdobj, nharm=2)$score
```

```
## VISUALIZE
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(embed, col=fk2$cluster, pch=19, main="K=2")
plot(embed, col=fk3$cluster, pch=19, main="K=3")
plot(embed, col=fk4$cluster, pch=19, main="K=4")
par(opar)
```

`gen3S` *Generate from Three 5-dimensional Subspaces in 200-dimensional space.*

Description

Generate from Three 5-dimensional Subspaces in 200-dimensional space.

Usage

```
gen3S(n = 50, var = 0.3)
```

Arguments

`n`	the number of data points sampled from each subspace (default: 50).
`var`	degree of Gaussian noise (default: 0.3).

Value

a named list containing with :

data an $(3 * n \times 3)$ data matrix.

class length-$3 * n$ vector for class label.

References

Wang S, Yuan X, Yao T, Yan S, Shen J (2011). "Efficient Subspace Segmentation via Quadratic Programming." In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, 519–524.

Examples

```
## a toy example
tester = gen3S(n=100)
data   = tester$data
label  = tester$class
```

genDONUTS *Generate Nested Donuts*

Description

It generates nested *donuts*, which are just hollow circles. For flexible testing, the parameter `k` controls the number of circles of varying radii where `n` controls the number of observations for each circle.

Usage

```
genDONUTS(n = 50, k = 2, sd = 0.1)
```

Arguments

n	the number of data points for each hollow circle (default: 50).
k	the number of circles (default: 2).
sd	magnitude of white noise (default: 0.1).

Value

a named list containing with $m = nk$:

data an $(m \times 2)$ data matrix.

label a length-m vector(factor) for class labels.

Examples

```
## generate data
donut2 = genDONUTS(k=2)
donut3 = genDONUTS(k=3)
donut4 = genDONUTS(k=4)

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3), pty="s")
plot(donut2$data, col=donut2$label, pch=19, main="k=2")
plot(donut3$data, col=donut3$label, pch=19, main="k=3")
plot(donut4$data, col=donut4$label, pch=19, main="k=4")
par(opar)
```

genLP *Generate Line and Plane Example with Fixed Number of Components*

Description

This function generates a toy example of 'line and plane' data in R^3 that data are generated from a mixture of lines (one-dimensional) planes (two-dimensional). The number of line- and plane-components are explicitly set by the user for flexible testing.

Usage

```
genLP(n = 100, nl = 1, np = 1, iso.var = 0.1)
```

Arguments

`n`	the number of data points for each line and plane.
`nl`	the number of line components.
`np`	the number of plane components.
`iso.var`	degree of isotropic variance.

Value

a named list containing with $m = n \times (nl + np)$:

data an $(m \times 3)$ data matrix.

class length-m vector for class label.

dimension length-m vector of corresponding dimension from which an observation is created.

Examples

```
## test for visualization
set.seed(10)
tester = genLP(n=100, nl=1, np=2, iso.var=0.1)
data   = tester$data
label  = tester$class

## do PCA for data reduction
proj = base::eigen(stats::cov(data))$vectors[,1:2]
dat2 = data%*%proj

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2), pty="s")
plot(dat2[,1],dat2[,2],pch=19,cex=0.5,col=label,main="PCA")
plot(data[,1],data[,2],pch=19,cex=0.5,col=label,main="Axis 1 vs 2")
plot(data[,1],data[,3],pch=19,cex=0.5,col=label,main="Axis 1 vs 3")
plot(data[,2],data[,3],pch=19,cex=0.5,col=label,main="Axis 2 vs 3")
par(opar)
```

```
## Not run:
## visualize in 3d
x11()
scatterplot3d::scatterplot3d(x=data, pch=19, cex.symbols=0.5, color=label)

## End(Not run)
```

`genSMILEY` *Generate SMILEY Data*

Description

Creates a smiley-face data in $\mathbf{R}^2$. This function is a modification of **mlbench**'s `mlbench.smiley` function.

Usage

```
genSMILEY(n = 496, sd = 0.1)
```

Arguments

`n`	number of samples to be generated.
`sd`	additive Gaussian noise level.

Value

a list containing

data an $(n \times 2)$ data matrix.

label a length-n vector(factor) for class labels.

Examples

```
## Generate SMILEY Data with Difference Noise Levels
s10 = genSMILEY(200, sd=0.1)
s25 = genSMILEY(200, sd=0.25)
s50 = genSMILEY(200, sd=0.5)

## Visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3), pty="s")
plot(s10$data, col=s10$label, pch=19, main="sd=0.10")
plot(s25$data, col=s25$label, pch=19, main="sd=0.25")
plot(s50$data, col=s50$label, pch=19, main="sd=0.50")
par(opar)
```

gmm *Finite Gaussian Mixture Model*

Description

Finite Gaussian Mixture Model (GMM) is a well-known probabilistic clustering algorithm by fitting the following distribution to the data

$$f(x; \{\mu_k, \Sigma_k\}_{k=1}^K) = \sum_{k=1}^K w_k N(x; \mu_k, \Sigma_k)$$

with parameters w_k's for cluster weights, μ_k's for class means, and Σ_k's for class covariances. This function is a wrapper for **Armadillo**'s GMM function, which supports two types of covariance models.

Usage

```
gmm(data, k = 2, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).
`...`	extra parameters including **maxiter** the maximum number of iterations (default: 10). **usediag** a logical; covariances are diagonal if `TRUE`, or full covariances are returned for `FALSE` (default: `FALSE`).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

mean a $(k \times p)$ matrix where each row is a class mean.

variance a $(p \times p \times k)$ array where each slice is a class covariance.

weight a length-k vector of class weights that sum to 1.

loglkd log-likelihood of the data for the fitted model.

algorithm name of the algorithm.

Examples

```
# -------------------------------------------------------------
#            clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = gmm(X, k=2)$cluster
cl3 = gmm(X, k=3)$cluster
cl4 = gmm(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="gmm: k=2")
plot(X2d, col=cl3, pch=19, main="gmm: k=3")
plot(X2d, col=cl4, pch=19, main="gmm: k=4")
par(opar)
```

`gmm03F` *Ensemble of Gaussian Mixtures with Random Projection*

Description

When the data lies in a high-dimensional Euclidean space, fitting a model-based clustering algorithm is troublesome. This function implements an algorithm from the reference, which uses an aggregate information from an ensemble of Gaussian mixtures in combination with random projection.

Usage

```
gmm03F(data, k = 2, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).
`...`	extra parameters including **nruns** the number of projections (default: 20). **lowdim** target dimension for random projection (default: 5).

maxiter the maximum number of iterations (default: 10).

usediag a logical; covariances are diagonal if `TRUE`, or full covariances are returned for `FALSE` (default: `FALSE`).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1 : k$).

algorithm name of the algorithm.

References

Fern XZ, Brodley CE (2003). "Random Projection for High Dimensional Data Clustering: A Cluster Ensemble Approach." In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, 186–193. ISBN 1577351894.

Examples

```
# -------------------------------------------------------------
#           clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = gmm03F(X, k=2)$cluster
cl3 = gmm03F(X, k=3)$cluster
cl4 = gmm03F(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,2), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="gmm03F: k=2")
plot(X2d, col=cl3, pch=19, main="gmm03F: k=3")
plot(X2d, col=cl4, pch=19, main="gmm03F: k=4")
par(opar)
```

gmm11R — *Regularized GMM by Ruan et al. (2011)*

Description

Ruan et al. (2011) proposed a regularized covariance estimation by graphical lasso to cope with high-dimensional scenario where conventional GMM might incur singular covariance components. Authors proposed to use λ as a regularization parameter as normally used in sparse covariance/precision estimation problems and suggested to use the model with the smallest BIC values.

Usage

```
gmm11R(data, k = 2, lambda = 1, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).
`lambda`	regularization parameter for graphical lasso (default: 1).
`...`	extra parameters including **maxiter** the maximum number of iterations (default: 10). **nstart** the number of random initializations (default: 5). **usediag** a logical; covariances are diagonal if `TRUE`, or full covariances are returned for `FALSE` (default: `FALSE`).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1 : k$).

mean a $(k \times p)$ matrix where each row is a class mean.

variance a $(p \times p \times k)$ array where each slice is a class covariance.

weight a length-k vector of class weights that sum to 1.

loglkd log-likelihood of the data for the fitted model.

algorithm name of the algorithm.

References

Ruan L, Yuan M, Zou H (2011). “Regularized Parameter Estimation in High-Dimensional Gaussian Mixture Models.” *Neural Computation*, **23**(6), 1605–1622. ISSN 0899-7667, 1530-888X.

Examples

```
# -------------------------------------------------------------
#             clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## COMPARE WITH STANDARD GMM
cl.gmm = gmm(X, k=3)$cluster
cl.11Rf = gmm11R(X, k=3)$cluster
cl.11Rd = gmm11R(X, k=3, usediag=TRUE)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3), pty="s")
plot(X2d, col=cl.gmm,  pch=19, main="standard GMM")
plot(X2d, col=cl.11Rf, pch=19, main="gmm11R: full covs")
plot(X2d, col=cl.11Rd, pch=19, main="gmm11R: diagonal covs")
par(opar)
```

gmm16G *Weighted GMM by Gebru et al. (2016)*

Description

When each observation x_i is associated with a weight $w_i > 0$, modifying the GMM formulation is required. Gebru et al. (2016) proposed a method to use scaled covariance based on an observation that

$$\mathcal{N}\left(x|\mu,\Sigma\right)^w \propto \mathcal{N}\left(x|\mu,\frac{\Sigma}{w}\right)$$

by considering the positive weight as a role of precision. Currently, we provide a method with fixed weight case only while the paper also considers a Bayesian formalism on the weight using Gamma distribution.

Usage

```
gmm16G(data, k = 2, weight = NULL, ...)
```

Arguments

data	an $(n \times p)$ matrix of row-stacked observations.
k	the number of clusters (default: 2).

`weight`	a positive weight vector of length n. If `NULL` (default), uniform weight is set.
`...`	extra parameters including **maxiter** the maximum number of iterations (default: 10). **usediag** a logical; covariances are diagonal if `TRUE`, or full covariances are returned for `FALSE` (default: `FALSE`).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

mean a $(k \times p)$ matrix where each row is a class mean.

variance a $(p \times p \times k)$ array where each slice is a class covariance.

weight a length-k vector of class weights that sum to 1.

loglkd log-likelihood of the data for the fitted model.

algorithm name of the algorithm.

References

Gebru ID, Alameda-Pineda X, Forbes F, Horaud R (2016). "EM Algorithms for Weighted-Data Clustering with Application to Audio-Visual Scene Analysis." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **38**(12), 2402–2415. ISSN 0162-8828, 2160-9292.

Examples

```
# -------------------------------------------------------------
#           clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = gmm16G(X, k=2)$cluster
cl3 = gmm16G(X, k=3)$cluster
cl4 = gmm16G(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="gmm16G: k=2")
plot(X2d, col=cl3, pch=19, main="gmm16G: k=3")
plot(X2d, col=cl4, pch=19, main="gmm16G: k=4")
par(opar)
```

gskmeans *Geodesic Spherical K-Means*

Description

Geodesic spherical k-means algorithm is an counterpart of the spherical k-means algorithm by replacing the cosine similarity with the squared geodesic distance, which is the great-circle distance under the intrinsic geometry regime on the unit hypersphere. If the data is not normalized, it performs the normalization and proceeds thereafter.

Usage

```
gskmeans(data, k = 2, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations. If not row-stochastic, each row is normalized to be unit norm.
`k`	the number of clusters (default: 2).
`...`	extra parameters including **init** initialization method; either `"kmeans"` or `"gmm"` (default: `"kmeans"`). **maxiter** the maximum number of iterations (default: 10). **abstol** stopping criterion to stop the algorithm (default: 10^{-8}). **verbose** a logical; `TRUE` to show iteration history or `FALSE` to quiet.

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1 : k$).

cost a value of the cost function.

means an $(k \times p)$ matrix where each row is a unit-norm class mean.

algorithm name of the algorithm.

Examples

```
# -------------------------------------------------------------
#            clustering with 'household' dataset
# -------------------------------------------------------------
## PREPARE
data(household, package="T4cluster")
X   = household$data
lab = as.integer(household$gender)

## EXECUTE GSKMEANS WITH VARYING K's
```

```
vec.rand = rep(0, 9)
for (i in 1:9){
  clust_i = gskmeans(X, k=(i+1))$cluster
  vec.rand[i] = compare.rand(clust_i, lab)
}

## VISUALIZE THE RAND INDEX
opar <- par(no.readonly=TRUE)
plot(2:10, vec.rand, type="b", pch=19, ylim=c(0.5, 1),
     ylab="Rand index",xlab="number of clusters",
     main="clustering quality index over varying k's.")
par(opar)
```

`household` *Load 'household' data*

Description

The data is taken from **HSAUR3** package's `household` data. We use housing, service, and food variables and normalize them to be unit-norm so that each observation is projected onto the 2-dimensional sphere. The data consists of 20 males and 20 females and has been used for clustering on the unit hypersphere.

Usage

```
data(household)
```

Format

a named list containing

data an $(n \times 3)$ data matrix whose rows are unit-norm.

gender a length-n factor for class label.

See Also

`household`

Examples

```
## Load the data
data(household, package="T4cluster")

## Visualize the data in pairs
opar <- par(no.readonly=TRUE)
scatterplot3d::scatterplot3d(household$data, color=rep(c("red","blue"), each=20),
              pch=19, main="household expenditure on the 2-dimensional sphere",
              xlim=c(0,1.2), ylim=c(0,1.2), zlim=c(0,1.2), angle=45)
par(opar)
```

kmeans *K-Means Clustering*

Description

K-means algorithm we provide is a wrapper to the **Armadillo**'s k-means routine. Two types of initialization schemes are employed. Please see the parameters section for more details.

Usage

```
kmeans(data, k = 2, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).
`...`	extra parameters including **init** initialization method; either `"random"` for random initialization, or `"plus"` for k-means++ starting. **maxiter** the maximum number of iterations (default: 10). **nstart** the number of random initializations (default: 5).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

mean a $(k \times p)$ matrix where each row is a class mean.

wcss within-cluster sum of squares (WCSS).

algorithm name of the algorithm.

References

Sanderson C, Curtin R (2016). “Armadillo: A Template-Based C++ Library for Linear Algebra.” *The Journal of Open Source Software*, **1**(2), 26. ISSN 2475-9066.

Examples

```
# -------------------------------------------------------------
#            clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = kmeans(X, k=2)$cluster
cl3 = kmeans(X, k=3)$cluster
cl4 = kmeans(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="k-means: k=2")
plot(X2d, col=cl3, pch=19, main="k-means: k=3")
plot(X2d, col=cl4, pch=19, main="k-means: k=4")
par(opar)
```

---

kmeans18B *K-Means Clustering with Lightweight Coreset*

Description

Apply k-means clustering algorithm on top of the lightweight coreset as proposed in the paper. The smaller the set is, the faster the execution becomes with potentially larger quantization errors.

Usage

```
kmeans18B(data, k = 2, m = round(nrow(data)/2), ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).
`m`	the size of coreset (default: $n/2$).
`...`	extra parameters including **maxiter** the maximum number of iterations (default: 10). **nstart** the number of random initializations (default: 5).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

mean a $(k \times p)$ matrix where each row is a class mean.

wcss within-cluster sum of squares (WCSS).

algorithm name of the algorithm.

References

Bachem O, Lucic M, Krause A (2018). "Scalable k -Means Clustering via Lightweight Coresets." In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery \& Data Mining*, 1119–1127. ISBN 978-1-4503-5552-0.

Examples

```
# -------------------------------------------------------------
#            clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT CORESET SIZES WITH K=3
core1 = kmeans18B(X, k=3, m=25)$cluster
core2 = kmeans18B(X, k=3, m=50)$cluster
core3 = kmeans18B(X, k=3, m=100)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=core1, pch=19, main="kmeans18B: m=25")
plot(X2d, col=core2, pch=19, main="kmeans18B: m=50")
plot(X2d, col=core3, pch=19, main="kmeans18B: m=100")
par(opar)
```

`kmeanspp` *K-Means++ Clustering*

Description

K-means++ algorithm is usually used as a fast initialization scheme, though it can still be used as a standalone clustering algorithms by first choosing the centroids and assign points to the nearest centroids.

Usage

```
kmeanspp(data, k = 2)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

algorithm name of the algorithm.

References

Arthur D, Vassilvitskii S (2007). "K-Means++: The Advantages of Careful Seeding." In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, 1027–1035. ISBN 978-0-89871-624-5.

Examples

```
# -------------------------------------------------------------
#            clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = kmeanspp(X, k=2)$cluster
cl3 = kmeanspp(X, k=3)$cluster
cl4 = kmeanspp(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="k-means++: k=2")
plot(X2d, col=cl3, pch=19, main="k-means++: k=3")
plot(X2d, col=cl4, pch=19, main="k-means++: k=4")
par(opar)
```

LRR *Low-Rank Representation*

Description

Low-Rank Representation (LRR) constructs the connectivity of the data by solving

$$\min_C \|C\|_* \quad \text{such that} \quad D = DC$$

for column-stacked data matrix D and $\|\cdot\|_*$ is the nuclear norm which is relaxation of the rank condition. If you are interested in full implementation of the algorithm with sparse outliers and noise, please contact the maintainer.

Usage

```
LRR(data, k = 2, rank = 2)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).
`rank`	sum of dimensions for all k subspaces (default: 2).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

algorithm name of the algorithm.

References

Liu G, Lin Z, Yu Y (2010). “Robust Subspace Segmentation by Low-Rank Representation.” In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML’10, 663–670. ISBN 978-1-60558-907-7.

Examples

```
## generate a toy example
set.seed(10)
tester = genLP(n=100, nl=2, np=1, iso.var=0.1)
data   = tester$data
label  = tester$class

## do PCA for data reduction
proj = base::eigen(stats::cov(data))$vectors[,1:2]
dat2 = data%*%proj
```

```
## run LRR algorithm with k=2, 3, and 4 with rank=4
output2 = LRR(data, k=2, rank=4)
output3 = LRR(data, k=3, rank=4)
output4 = LRR(data, k=4, rank=4)

## extract label information
lab2 = output2$cluster
lab3 = output3$cluster
lab4 = output4$cluster

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(dat2, pch=19, cex=0.9, col=lab2, main="LRR:K=2")
plot(dat2, pch=19, cex=0.9, col=lab3, main="LRR:K=3")
plot(dat2, pch=19, cex=0.9, col=lab4, main="LRR:K=4")
par(opar)
```

`LRSC` *Low-Rank Subspace Clustering*

Description

Low-Rank Subspace Clustering (LRSC) constructs the connectivity of the data by solving

$$\min_C \|C\|_* \quad \text{such that} \quad A = AC,\ C = C^\top$$

for the uncorrupted data scenario where A is a column-stacked data matrix. In the current implementation, the first equality constraint for reconstructiveness of the data can be relaxed by solving

$$\min_C \|C\|_* + \frac{\tau}{2}\|A - AC\|_F^2 \quad \text{such that} \quad C = C^\top$$

controlled by the regularization parameter τ. If you are interested in enabling a more general class of the problem suggested by authors, please contact maintainer of the package.

Usage

```
LRSC(data, k = 2, type = c("relaxed", "exact"), tau = 1)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).
`type`	type of the problem to be solved.
`tau`	regularization parameter for relaxed-constraint problem.

Details

$$\min_C \|C\|_* \quad \text{such that} \quad D = DC$$

for column-stacked data matrix D and $\|\cdot\|_*$ is the nuclear norm which is relaxation of the rank condition. If you are interested in full implementation of the algorithm with sparse outliers and noise, please contact the maintainer.

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

algorithm name of the algorithm.

References

Vidal R, Favaro P (2014). "Low Rank Subspace Clustering (LRSC)." *Pattern Recognition Letters*, **43**, 47–61. ISSN 01678655.

Examples

```
## generate a toy example
set.seed(10)
tester = genLP(n=100, nl=2, np=1, iso.var=0.1)
data   = tester$data
label  = tester$class

## do PCA for data reduction
proj = base::eigen(stats::cov(data))$vectors[,1:2]
dat2 = data%*%proj

## run LRSC algorithm with k=2,3,4 with relaxed/exact solvers
out2rel = LRSC(data, k=2, type="relaxed")
out3rel = LRSC(data, k=3, type="relaxed")
out4rel = LRSC(data, k=4, type="relaxed")

out2exc = LRSC(data, k=2, type="exact")
out3exc = LRSC(data, k=3, type="exact")
out4exc = LRSC(data, k=4, type="exact")

## extract label information
lab2rel = out2rel$cluster
lab3rel = out3rel$cluster
lab4rel = out4rel$cluster

lab2exc = out2exc$cluster
lab3exc = out3exc$cluster
lab4exc = out4exc$cluster

## visualize
```

```
opar <- par(no.readonly=TRUE)
par(mfrow=c(2,3))
plot(dat2, pch=19, cex=0.9, col=lab2rel, main="LRSC Relaxed:K=2")
plot(dat2, pch=19, cex=0.9, col=lab3rel, main="LRSC Relaxed:K=3")
plot(dat2, pch=19, cex=0.9, col=lab4rel, main="LRSC Relaxed:K=4")
plot(dat2, pch=19, cex=0.9, col=lab2exc, main="LRSC Exact:K=2")
plot(dat2, pch=19, cex=0.9, col=lab3exc, main="LRSC Exact:K=3")
plot(dat2, pch=19, cex=0.9, col=lab4exc, main="LRSC Exact:K=4")
par(opar)
```

`LSR` *Least Squares Regression*

Description

For the subspace clustering, traditional method of least squares regression is used to build coefficient matrix that reconstructs the data point by solving

$$\textrm{min}_Z \|X - XZ\|_F^2 + \lambda \|Z\|_F \textrm{ such that } diag(Z) = 0$$

where $X \in \mathbf{R}^{p \times n}$ is a column-stacked data matrix. As seen from the equation, we use a denoising version controlled by λ and provide an option to abide by the constraint $diag(Z) = 0$ by `zerodiag` parameter.

Usage

```
LSR(data, k = 2, lambda = 1e-05, zerodiag = TRUE)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).
`lambda`	regularization parameter (default: 1e-5).
`zerodiag`	a logical; `TRUE` (default) to use the problem formulation with zero diagonal entries or `FALSE` otherwise.

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1 : k$).

algorithm name of the algorithm.

References

Lu C, Min H, Zhao Z, Zhu L, Huang D, Yan S (2012). "Robust and Efficient Subspace Segmentation via Least Squares Regression." In Hutchison D, Kanade T, Kittler J, Kleinberg JM, Mattern F, Mitchell JC, Naor M, Nierstrasz O, Pandu Rangan C, Steffen B, Sudan M, Terzopoulos D, Tygar D, Vardi MY, Weikum G, Fitzgibbon A, Lazebnik S, Perona P, Sato Y, Schmid C (eds.), *Computer Vision -ECCV 2012*, volume 7578, 347–360. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-33785-7 978-3-642-33786-4.

Examples

```
## generate a toy example
set.seed(10)
tester = genLP(n=100, nl=2, np=1, iso.var=0.1)
data   = tester$data
label  = tester$class

## do PCA for data reduction
proj = base::eigen(stats::cov(data))$vectors[,1:2]
dat2 = data%*%proj

## run LSR for k=3 with different lambda values
out1 = LSR(data, k=3, lambda=1e-2)
out2 = LSR(data, k=3, lambda=1)
out3 = LSR(data, k=3, lambda=1e+2)

## extract label information
lab1 = out1$cluster
lab2 = out2$cluster
lab3 = out3$cluster

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(dat2, pch=19, cex=0.9, col=lab1, main="LSR:lambda=1e-2")
plot(dat2, pch=19, cex=0.9, col=lab2, main="LSR:lambda=1")
plot(dat2, pch=19, cex=0.9, col=lab3, main="LSR:lambda=1e+2")
par(opar)
```

MSM *Bayesian Mixture of Subspaces of Different Dimensions*

Description

MSM is a Bayesian model inferring mixtures of subspaces that are of possibly different dimensions. For simplicity, this function returns only a handful of information that are most important in representing the mixture model, including projection, location, and hard assignment parameters.

Usage

```
MSM(data, k = 2, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of mixtures.
`...`	extra parameters including **temperature** temperature value for Gibbs posterior (default: 1e-6). **prop.var** proposal variance parameter (default: 1.0). **iter** the number of MCMC runs (default: 496). **burn.in** burn-in for MCMC runs (default: iter/2). **thin** interval for recording MCMC runs (default: 10). **print.progress** a logical; `TRUE` to show completion of iterations by 10, `FALSE` otherwise (default: `FALSE`).

Value

a list whose elements are S3 class `"MSM"` instances, which are also lists of following elements:

P length-`k` list of projection matrices.

U length-`k` list of orthonormal basis.

theta length-`k` list of center locations of each mixture.

cluster length-`n` vector of cluster label.

Examples

```
## generate a toy example
set.seed(10)
tester = genLP(n=100, nl=2, np=1, iso.var=0.1)
data   = tester$data
label  = tester$class

## do PCA for data reduction
proj = base::eigen(stats::cov(data))$vectors[,1:2]
dat2 = data%*%proj

## run MSM algorithm with k=2, 3, and 4
maxiter = 500
output2 = MSM(data, k=2, iter=maxiter)
output3 = MSM(data, k=3, iter=maxiter)
output4 = MSM(data, k=4, iter=maxiter)

## extract final clustering information
nrec  = length(output2)
finc2 = output2[[nrec]]$cluster
finc3 = output3[[nrec]]$cluster
```

```
finc4 = output4[[nrec]]$cluster

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(3,4))
plot(dat2[,1],dat2[,2],pch=19,cex=0.3,col=finc2+1,main="K=2:PCA")
plot(data[,1],data[,2],pch=19,cex=0.3,col=finc2+1,main="K=2:Axis(1,2)")
plot(data[,1],data[,3],pch=19,cex=0.3,col=finc2+1,main="K=2:Axis(1,3)")
plot(data[,2],data[,3],pch=19,cex=0.3,col=finc2+1,main="K=2:Axis(2,3)")

plot(dat2[,1],dat2[,2],pch=19,cex=0.3,col=finc3+1,main="K=3:PCA")
plot(data[,1],data[,2],pch=19,cex=0.3,col=finc3+1,main="K=3:Axis(1,2)")
plot(data[,1],data[,3],pch=19,cex=0.3,col=finc3+1,main="K=3:Axis(1,3)")
plot(data[,2],data[,3],pch=19,cex=0.3,col=finc3+1,main="K=3:Axis(2,3)")

plot(dat2[,1],dat2[,2],pch=19,cex=0.3,col=finc4+1,main="K=4:PCA")
plot(data[,1],data[,2],pch=19,cex=0.3,col=finc4+1,main="K=4:Axis(1,2)")
plot(data[,1],data[,3],pch=19,cex=0.3,col=finc4+1,main="K=4:Axis(1,3)")
plot(data[,2],data[,3],pch=19,cex=0.3,col=finc4+1,main="K=4:Axis(2,3)")
par(opar)
```

pcm — *Compute Pairwise Co-occurrence Matrix*

Description

Let *clustering* be a label from data of N observations and suppose we are given M such labels. Co-occurrent matrix counts the number of events where two observations X_i and X_j belong to the same category/class. *PCM* serves as a measure of uncertainty embedded in any algorithms with non-deterministic components.

Usage

```
pcm(partitions)
```

Arguments

`partitions`	partitions can be provided in either (1) an $(M \times N)$ matrix where each row is a clustering for N objects, or (2) a length-M list of length-N clustering labels.

Value

an $(N \times N)$ matrix, whose elements (i, j) are counts for how many times observations i and j belong to the same cluster, ranging from 0 to M.

See Also

psm

Examples

```
# -------------------------------------------------------------
#              PSM with 'iris' dataset + k-means++
# -------------------------------------------------------------
## PREPARE WITH SUBSET OF DATA
data(iris)
X     = as.matrix(iris[,1:4])
lab   = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## RUN K-MEANS++ 100 TIMES
partitions = list()
for (i in 1:100){
  partitions[[i]] = kmeanspp(X)$cluster
}

## COMPUTE PCM
iris.pcm = pcm(partitions)

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
image(iris.pcm[,150:1], axes=FALSE, main="PCM")
par(opar)
```

predict.MSM *S3 method to predict class label of new data with 'MSM' object*

Description

Given an instance of MSM class from MSM function, predict class label of a new data.

Usage

```
## S3 method for class 'MSM'
predict(object, newdata, ...)
```

Arguments

object	an 'MSM' object from MSM function.
newdata	an $(m \times p)$ matrix of row-stacked observations.
...	extra parameters (not necessary).

Value

a length-m vector of class labels.

psm *Compute Posterior Similarity Matrix*

Description

Let *clustering* be a label from data of N observations and suppose we are given M such labels. Posterior similarity matrix, as its name suggests, computes posterior probability for a pair of observations to belong to the same cluster, i.e.,

$$P_{ij} = P(\text{label}(X_i) = \text{label}(X_j))$$

under the scenario where multiple clusterings are samples drawn from a posterior distribution within the Bayesian framework. However, it can also be used for non-Bayesian settings as `psm` is a measure of uncertainty embedded in any algorithms with non-deterministic components.

Usage

```
psm(partitions)
```

Arguments

`partitions`	partitions can be provided in either (1) an $(M \times N)$ matrix where each row is a clustering for N objects, or (2) a length-M list of length-N clustering labels.

Value

an $(N \times N)$ matrix, whose elements (i, j) are posterior probability for an observation i and j belong to the same cluster.

See Also

`pcm`

Examples

```
# -------------------------------------------------------------
#              PSM with 'iris' dataset + k-means++
# -------------------------------------------------------------
## PREPARE WITH SUBSET OF DATA
data(iris)
X     = as.matrix(iris[,1:4])
lab   = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## RUN K-MEANS++ 100 TIMES
partitions = list()
for (i in 1:100){
  partitions[[i]] = kmeanspp(X)$cluster
}

## COMPUTE PSM
iris.psm = psm(partitions)

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,2), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
image(iris.psm[,150:1], axes=FALSE, main="PSM")
par(opar)
```

---

sc05Z *Spectral Clustering by Zelnik-Manor and Perona (2005)*

Description

Zelnik-Manor and Perona proposed a method to define a set of data-driven bandwidth parameters where σ_i is the distance from a point x_i to its `nnbd`-th nearest neighbor. Then the affinity matrix is defined as

$$A_{ij} = \exp(-d(x_i, d_j)^2/\sigma_i\sigma_j)$$

and the standard spectral clustering of Ng, Jordan, and Weiss (`scNJW`) is applied.

Usage

```
sc05Z(data, k = 2, nnbd = 7, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations or S3 `dist` object of n observations.
`k`	the number of clusters (default: 2).
`nnbd`	neighborhood size to define data-driven bandwidth parameter (default: 7).
`...`	extra parameters including **algclust** method to perform clustering on embedded data; either `"kmeans"` (default) or `"GMM"`. **maxiter** the maximum number of iterations (default: 10).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

eigval eigenvalues of the graph laplacian's spectral decomposition.

embeds an $(n \times k)$ low-dimensional embedding.

algorithm name of the algorithm.

References

Zelnik-manor L, Perona P (2005). “Self-Tuning Spectral Clustering.” In Saul LK, Weiss Y, Bottou L (eds.), *Advances in Neural Information Processing Systems 17*, 1601–1608. MIT Press.

Examples

```
# -------------------------------------------------------------
#            clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = sc05Z(X, k=2)$cluster
cl3 = sc05Z(X, k=3)$cluster
cl4 = sc05Z(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="sc05Z: k=2")
plot(X2d, col=cl3, pch=19, main="sc05Z: k=3")
plot(X2d, col=cl4, pch=19, main="sc05Z: k=4")
par(opar)
```

---

sc09G *Spectral Clustering by Gu and Wang (2009)*

Description

The algorithm defines a set of data-driven bandwidth parameters where σ_i is the average distance from a point x_i to its nnbd-th nearest neighbor. Then the affinity matrix is defined as

$$A_{ij} = \exp(-d(x_i, d_j)^2/\sigma_i\sigma_j)$$

and the standard spectral clustering of Ng, Jordan, and Weiss (`scNJW`) is applied.

Usage

```
sc09G(data, k = 2, nnbd = 7, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations or S3 `dist` object of n observations.
`k`	the number of clusters (default: 2).
`nnbd`	neighborhood size to define data-driven bandwidth parameter (default: 7).
`...`	extra parameters including **algclust** method to perform clustering on embedded data; either `"kmeans"` (default) or `"GMM"`. **maxiter** the maximum number of iterations (default: 10).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1 : k$).

eigval eigenvalues of the graph laplacian's spectral decomposition.

embeds an $(n \times k)$ low-dimensional embedding.

algorithm name of the algorithm.

References

Gu R, Wang J (2009). "An Improved Spectral Clustering Algorithm Based on Neighbour Adaptive Scale." In *2009 International Conference on Business Intelligence and Financial Engineering*, 233–236. ISBN 978-0-7695-3705-4.

Examples

```
# -------------------------------------------------------------
#           clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = sc09G(X, k=2)$cluster
cl3 = sc09G(X, k=3)$cluster
cl4 = sc09G(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="sc09G: k=2")
plot(X2d, col=cl3, pch=19, main="sc09G: k=3")
plot(X2d, col=cl4, pch=19, main="sc09G: k=4")
par(opar)
```

`sc10Z` *Spectral Clustering by Zhang et al. (2010)*

Description

The algorithm defines a set of data-driven bandwidth parameters p_{ij} by constructing a similarity matrix. Then the affinity matrix is defined as

$$A_{ij} = \exp(-d(x_i, d_j)^2 / 2p_{ij})$$

and the standard spectral clustering of Ng, Jordan, and Weiss (`scNJW`) is applied.

Usage

```
sc10Z(data, k = 2, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations or S3 `dist` object of n observations.
`k`	the number of clusters (default: 2).
`...`	extra parameters including **algclust** method to perform clustering on embedded data; either `"kmeans"` (default) or `"GMM"`. **maxiter** the maximum number of iterations (default: 10).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

eigval eigenvalues of the graph laplacian's spectral decomposition.

embeds an $(n \times k)$ low-dimensional embedding.

algorithm name of the algorithm.

References

Zhang Y, Zhou J, Fu Y (2010). "Spectral Clustering Algorithm Based on Adaptive Neighbor Distance Sort Order." In *The 3rd International Conference on Information Sciences and Interaction Sciences*, 444–447. ISBN 978-1-4244-7384-7.

Examples

```
# -------------------------------------------------------------
#            clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = sc10Z(X, k=2)$cluster
cl3 = sc10Z(X, k=3)$cluster
cl4 = sc10Z(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="sc10Z: k=2")
plot(X2d, col=cl3, pch=19, main="sc10Z: k=3")
plot(X2d, col=cl4, pch=19, main="sc10Z: k=4")
par(opar)
```

---

sc11Y *Spectral Clustering by Yang et al. (2011)*

Description

As a data-driven method, the algorithm recovers geodesic distance from a k-nearest neighbor graph scaled by an (exponential) parameter ρ and applies random-walk spectral clustering. Authors referred their method as density sensitive similarity function.

Usage

```
sc11Y(data, k = 2, nnbd = 7, rho = 2, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations or S3 `dist` object of n observations.
`k`	the number of clusters (default: 2).
`nnbd`	neighborhood size to define data-driven bandwidth parameter (default: 7).
`rho`	exponent scaling parameter (default: 2).
`...`	extra parameters including

algclust method to perform clustering on embedded data; either "kmeans" (default) or "GMM".

maxiter the maximum number of iterations (default: 10).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

eigval eigenvalues of the graph laplacian's spectral decomposition.

embeds an $(n \times k)$ low-dimensional embedding.

algorithm name of the algorithm.

References

Yang P, Zhu Q, Huang B (2011). "Spectral Clustering with Density Sensitive Similarity Function." *Knowledge-Based Systems*, **24**(5), 621–628. ISSN 09507051.

Examples

```
# -------------------------------------------------------------
#            clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = sc11Y(X, k=2)$cluster
cl3 = sc11Y(X, k=3)$cluster
cl4 = sc11Y(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="sc11Y: k=2")
plot(X2d, col=cl3, pch=19, main="sc11Y: k=3")
plot(X2d, col=cl4, pch=19, main="sc11Y: k=4")
par(opar)
```

`sc12L` *Spectral Clustering by Li and Guo (2012)*

Description

Li and Guo proposed to construct an affinity matrix

$$A_{ij} = \exp(-d(x_i, d_j)^2 / 2\sigma^2)$$

and adjust the matrix by neighbor propagation. Then, standard spectral clustering from the symmetric, normalized graph laplacian is applied.

Usage

```
sc12L(data, k = 2, sigma = 1, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations or S3 `dist` object of n observations.
`k`	the number of clusters (default: 2).
`sigma`	common bandwidth parameter (default: 1).
`...`	extra parameters including **algclust** method to perform clustering on embedded data; either `"kmeans"` (default) or `"GMM"`. **maxiter** the maximum number of iterations (default: 10).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

eigval eigenvalues of the graph laplacian's spectral decomposition.

embeds an $(n \times k)$ low-dimensional embedding.

algorithm name of the algorithm.

References

Li X, Guo L (2012). "Constructing Affinity Matrix in Spectral Clustering Based on Neighbor Propagation." *Neurocomputing*, **97**, 125–130. ISSN 09252312.

See Also

`scNJW`

Examples

```
# -------------------------------------------------------------
#            clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = sc12L(X, k=2)$cluster
cl3 = sc12L(X, k=3)$cluster
cl4 = sc12L(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="sc12L: k=2")
plot(X2d, col=cl3, pch=19, main="sc12L: k=3")
plot(X2d, col=cl4, pch=19, main="sc12L: k=4")
par(opar)
```

---

scNJW   *Spectral Clustering by Ng, Jordan, and Weiss (2002)*

Description

The version of Ng, Jordan, and Weiss first constructs the affinity matrix

$$A_{ij} = \exp(-d(x_i, d_j)^2/\sigma^2)$$

where σ is a common bandwidth parameter and performs k-means (or possibly, GMM) clustering on the row-space of eigenvectors for the symmetric graph laplacian matrix

$$L = D^{-1/2}(D - A)D^{-1/2}$$

.

Usage

```
scNJW(data, k = 2, sigma = 1, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations or S3 `dist` object of n observations.
`k`	the number of clusters (default: 2).
`sigma`	bandwidth parameter (default: 1).
`...`	extra parameters including **algclust** method to perform clustering on embedded data; either `"kmeans"` (default) or `"GMM"`. **maxiter** the maximum number of iterations (default: 10).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

eigval eigenvalues of the graph laplacian's spectral decomposition.

embeds an $(n \times k)$ low-dimensional embedding.

algorithm name of the algorithm.

References

Ng AY, Jordan MI, Weiss Y (2002). "On Spectral Clustering: Analysis and an Algorithm." In Dietterich TG, Becker S, Ghahramani Z (eds.), *Advances in Neural Information Processing Systems 14*, 849–856. MIT Press.

Examples

```
# -------------------------------------------------------------
#            clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = scNJW(X, k=2)$cluster
cl3 = scNJW(X, k=3)$cluster
cl4 = scNJW(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="scNJW: k=2")
plot(X2d, col=cl3, pch=19, main="scNJW: k=3")
plot(X2d, col=cl4, pch=19, main="scNJW: k=4")
par(opar)
```

`scSM` *Spectral Clustering by Shi and Malik (2000)*

Description

The version of Shi and Malik first constructs the affinity matrix

$$A_{ij} = \exp(-d(x_i, d_j)^2/\sigma^2)$$

where σ is a common bandwidth parameter and performs k-means (or possibly, GMM) clustering on the row-space of eigenvectors for the random-walk graph laplacian matrix

$$L = D^{-1}(D - A)$$

.

Usage

```
scSM(data, k = 2, sigma = 1, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations or S3 `dist` object of n observations.
`k`	the number of clusters (default: 2).
`sigma`	bandwidth parameter (default: 1).
`...`	extra parameters including **algclust** method to perform clustering on embedded data; either `"kmeans"` (default) or `"GMM"`. **maxiter** the maximum number of iterations (default: 10).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1 : k$).

eigval eigenvalues of the graph laplacian's spectral decomposition.

embeds an $(n \times k)$ low-dimensional embedding.

algorithm name of the algorithm.

References

Shi J, Malik J (Aug./2000). "Normalized Cuts and Image Segmentation." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(8), 888–905. ISSN 01628828.

Examples

```
# -----------------------------------------------------------
#           clustering with 'iris' dataset
# -----------------------------------------------------------
## PREPARE WITH SUBSET OF DATA
data(iris)
sid = sample(1:150, 50)
X   = as.matrix(iris[sid,1:4])
lab = as.integer(as.factor(iris[sid,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = scSM(X, k=2)$cluster
cl3 = scSM(X, k=3)$cluster
cl4 = scSM(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="scSM: k=2")
plot(X2d, col=cl3, pch=19, main="scSM: k=3")
plot(X2d, col=cl4, pch=19, main="scSM: k=4")
par(opar)
```

scUL *Spectral Clustering with Unnormalized Laplacian*

Description

The version of Shi and Malik first constructs the affinity matrix

$$A_{ij} = \exp(-d(x_i, d_j)^2/\sigma^2)$$

where σ is a common bandwidth parameter and performs k-means (or possibly, GMM) clustering on the row-space of eigenvectors for the unnormalized graph laplacian matrix

$$L = D - A$$

.

Usage

```
scUL(data, k = 2, sigma = 1, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations or S3 `dist` object of n observations.
`k`	the number of clusters (default: 2).
`sigma`	bandwidth parameter (default: 1).
`...`	extra parameters including **algclust** method to perform clustering on embedded data; either `"kmeans"` (default) or `"GMM"`. **maxiter** the maximum number of iterations (default: 10).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

eigval eigenvalues of the graph laplacian's spectral decomposition.

embeds an $(n \times k)$ low-dimensional embedding.

algorithm name of the algorithm.

References

von Luxburg U (2007). "A Tutorial on Spectral Clustering." *Statistics and Computing*, **17**(4), 395–416. ISSN 0960-3174, 1573-1375.

Examples

```
# -------------------------------------------------------------
#            clustering with 'iris' dataset
# -------------------------------------------------------------
## PREPARE
data(iris)
X   = as.matrix(iris[,1:4])
lab = as.integer(as.factor(iris[,5]))

## EMBEDDING WITH PCA
X2d = Rdimtools::do.pca(X, ndim=2)$Y

## CLUSTERING WITH DIFFERENT K VALUES
cl2 = scUL(X, k=2)$cluster
cl3 = scUL(X, k=3)$cluster
cl4 = scUL(X, k=4)$cluster

## VISUALIZATION
opar <- par(no.readonly=TRUE)
```

```
par(mfrow=c(1,4), pty="s")
plot(X2d, col=lab, pch=19, main="true label")
plot(X2d, col=cl2, pch=19, main="scUL: k=2")
plot(X2d, col=cl3, pch=19, main="scUL: k=3")
plot(X2d, col=cl4, pch=19, main="scUL: k=4")
par(opar)
```

`spkmeans` *Spherical K-Means Clustering*

Description

Spherical k-means algorithm performs clustering for the data residing on the unit hypersphere with the cosine similarity. If the data is not normalized, it performs the normalization and proceeds thereafter.

Usage

```
spkmeans(data, k = 2, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations. If not row-stochastic, each row is normalized to be unit norm.
`k`	the number of clusters (default: 2).
`...`	extra parameters including **init** initialization method; either `"kmeans"` or `"gmm"` (default: `"kmeans"`). **maxiter** the maximum number of iterations (default: 10). **abstol** stopping criterion to stop the algorithm (default: 10^{-8}). **verbose** a logical; `TRUE` to show iteration history or `FALSE` to quiet.

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1 : k$).

cost a value of the cost function.

means an $(k \times p)$ matrix where each row is a unit-norm class mean.

algorithm name of the algorithm.

References

I. S. Dhillon and D. S. Modha (2001). "Concept decompositions for large sparse text data using clustering." *Machine Learning*, **42**:143–175.

Examples

```
# -------------------------------------------------------------
#            clustering with 'household' dataset
# -------------------------------------------------------------
## PREPARE
data(household, package="T4cluster")
X   = household$data
lab = as.integer(household$gender)

## EXECUTE SPKMEANS WITH VARYING K's
vec.rand = rep(0, 9)
for (i in 1:9){
  clust_i = spkmeans(X, k=(i+1))$cluster
  vec.rand[i] = compare.rand(clust_i, lab)
}

## VISUALIZE THE RAND INDEX
opar <- par(no.readonly=TRUE)
plot(2:10, vec.rand, type="b", pch=19, ylim=c(0.5, 1),
     ylab="Rand index",xlab="number of clusters",
     main="clustering quality index over varying k's.")
par(opar)
```

SSC *Sparse Subspace Clustering*

Description

Sparse Subspace Clustering (SSC) assumes that the data points lie in a union of low-dimensional subspaces. The algorithm constructs local connectivity and uses the information for spectral clustering. `SSC` is an implementation based on basis pursuit for sparse reconstruction for the model without systematic noise, which solves

$$\textrm{min}_C \|C\|_1 \quad \textrm{such that} \quad diag(C) = 0,\ D = DC$$

for column-stacked data matrix D. If you are interested in full implementation of the algorithm with sparse outliers and noise, please contact the maintainer.

Usage

```
SSC(data, k = 2)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1 : k$).

algorithm name of the algorithm.

References

Elhamifar E, Vidal R (2009). “Sparse Subspace Clustering.” In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2790–2797. ISBN 978-1-4244-3992-8.

Examples

```
## generate a toy example
set.seed(10)
tester = genLP(n=100, nl=2, np=1, iso.var=0.1)
data   = tester$data
label  = tester$class

## do PCA for data reduction
proj = base::eigen(stats::cov(data))$vectors[,1:2]
dat2 = data%*%proj

## run SSC algorithm with k=2, 3, and 4
output2 = SSC(data, k=2)
output3 = SSC(data, k=3)
output4 = SSC(data, k=4)

## extract label information
lab2 = output2$cluster
lab3 = output3$cluster
lab4 = output4$cluster

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(3,4))
plot(dat2[,1],dat2[,2],pch=19,cex=0.3,col=lab2,main="K=2:PCA")
plot(data[,1],data[,2],pch=19,cex=0.3,col=lab2,main="K=2:Axis(1,2)")
plot(data[,1],data[,3],pch=19,cex=0.3,col=lab2,main="K=2:Axis(1,3)")
plot(data[,2],data[,3],pch=19,cex=0.3,col=lab2,main="K=2:Axis(2,3)")

plot(dat2[,1],dat2[,2],pch=19,cex=0.3,col=lab3,main="K=3:PCA")
plot(data[,1],data[,2],pch=19,cex=0.3,col=lab3,main="K=3:Axis(1,2)")
plot(data[,1],data[,3],pch=19,cex=0.3,col=lab3,main="K=3:Axis(1,3)")
plot(data[,2],data[,3],pch=19,cex=0.3,col=lab3,main="K=3:Axis(2,3)")

plot(dat2[,1],dat2[,2],pch=19,cex=0.3,col=lab4,main="K=4:PCA")
plot(data[,1],data[,2],pch=19,cex=0.3,col=lab4,main="K=4:Axis(1,2)")
plot(data[,1],data[,3],pch=19,cex=0.3,col=lab4,main="K=4:Axis(1,3)")
plot(data[,2],data[,3],pch=19,cex=0.3,col=lab4,main="K=4:Axis(2,3)")
par(opar)
```

`SSQP` *Subspace Segmentation via Quadratic Programming*

Description

Subspace Segmentation via Quadratic Programming (SSQP) solves the following problem

$$\min_Z \|X - XZ\|_F^2 + \lambda \|Z^\top Z\|_1 \text{ such that } diag(Z) = 0,\ Z \leq 0$$

where $X \in \mathbf{R}^{p\times n}$ is a column-stacked data matrix. The computed Z^* is used as an affinity matrix for spectral clustering.

Usage

```
SSQP(data, k = 2, lambda = 1e-05, ...)
```

Arguments

`data`	an $(n \times p)$ matrix of row-stacked observations.
`k`	the number of clusters (default: 2).
`lambda`	regularization parameter (default: 1e-5).
`...`	extra parameters for the gradient descent algorithm including **maxiter** maximum number of iterations (default: 100). **abstol** tolerance level to stop (default: 1e-7).

Value

a named list of S3 class `T4cluster` containing

cluster a length-n vector of class labels (from $1:k$).

algorithm name of the algorithm.

References

Wang S, Yuan X, Yao T, Yan S, Shen J (2011). “Efficient Subspace Segmentation via Quadratic Programming.” In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI’11, 519–524.

Examples

```
## generate a toy example
set.seed(10)
tester = genLP(n=100, nl=2, np=1, iso.var=0.1)
data   = tester$data
label  = tester$class

## do PCA for data reduction
proj = base::eigen(stats::cov(data))$vectors[,1:2]
dat2 = data%*%proj

## run SSQP for k=3 with different lambda values
out1 = SSQP(data, k=3, lambda=1e-2)
out2 = SSQP(data, k=3, lambda=1)
out3 = SSQP(data, k=3, lambda=1e+2)

## extract label information
lab1 = out1$cluster
lab2 = out2$cluster
lab3 = out3$cluster

## visualize
opar <- par(no.readonly=TRUE)
par(mfrow=c(1,3))
plot(dat2, pch=19, cex=0.9, col=lab1, main="SSQP:lambda=1e-2")
plot(dat2, pch=19, cex=0.9, col=lab2, main="SSQP:lambda=1")
plot(dat2, pch=19, cex=0.9, col=lab3, main="SSQP:lambda=1e+2")
par(opar)
```

Index

∗ **algorithm**
- dpmeans, 5
- gmm, 15
- gmm03F, 16
- gmm11R, 18
- gmm16G, 19
- kmeans, 23
- kmeans18B, 24
- kmeanspp, 25
- sc05Z, 36
- sc09G, 37
- sc10Z, 39
- sc11Y, 40
- sc12L, 42
- scNJW, 43
- scSM, 45
- scUL, 46

∗ **comparison**
- compare.adjrand, 2
- compare.rand, 4

∗ **datasets**
- household, 22

∗ **data**
- gen3S, 11
- genDONUTS, 12
- genLP, 13
- genSMILEY, 14
- household, 22

∗ **functional**
- funhclust, 8
- funkmeans03A, 9

∗ **soc**
- pcm, 33
- psm, 35

∗ **sphere**
- gskmeans, 21
- spkmeans, 48

∗ **subspace**
- EKSS, 6
- LRR, 27
- LRSC, 28
- LSR, 30
- MSM, 31
- predict.MSM, 34
- SSC, 49
- SSQP, 51

compare.adjrand, 2
compare.rand, *3*, 4

dpmeans, 5

EKSS, 6

funhclust, 8
funkmeans03A, 9

gen3S, 11
genDONUTS, 12
genLP, 13
genSMILEY, 14
gmm, 15
gmm03F, 16
gmm11R, 18
gmm16G, 19
gskmeans, 21

hclust, *8*
household, *22*, 22

kmeans, 23
kmeans18B, 24
kmeanspp, 25

LRR, 27
LRSC, 28
LSR, 30

MSM, 31, *34*

pcm, 33, *35*

predict.MSM, 34
psm, *33*, 35

sc05Z, 36
sc09G, 37
sc10Z, 39
sc11Y, 40
sc12L, 42
scNJW, *36*, *37*, *39*, *42*, 43
scSM, 45
scUL, 46
spkmeans, 48
SSC, 49
SSQP, 51