Package ‘pivmet’

October 2, 2018

Type Package

Title Pivotal methods for Bayesian relabelling and k-means clustering

Version 0.1.0

Author Leonardo Egidi

Maintainer Leonardo Egidi <legidi@units.it>

Description The package provides some pivotal algorithms
for dealing with Bayesian Gaussian mixture models and relabelling the
MCMC chains in order to undo the label switching problem.
The same pivotal methods may be used with for initialize the
centers of the classical k-means
algorithm in order to obtain a better clustering solution.

url https://github.com/leoegidi/pivmet

Encoding UTF-8

LazyData true

LazyLoad yes

Imports mvtnorm, bayesmix, RcmdrMisc,
cluster, mclust, runjags, rjags, MASS

Depends R, mvtnorm, bayesmix, RcmdrMisc,
cluster, mclust, runjags, rjags, MASS

Suggests knitr

VignetteBuilder knitr

RemoteType github

RemoteHost https://api.github.com

RemoteRepo pivmet

RemoteUsername LeoEgidi

RemoteRef master

RemoteSha 59e8616331302cc911ae13b6190580dcfee099b9

GithubRepo pivmet

GithubUsername LeoEgidi

GithubRef master

GithubSHA1 59e8616331302cc911ae13b6190580dcfee099b9

RoxygenNote 6.1.0

BuildManual yes

R topics documented:

`MUS` *MUS algorithm*

Description

Perform Maxima Units Search (MUS) algorithm on a large and sparse matrix in order to find a set of pivotal units through a sequential search in the given matrix.

Usage

```
MUS(C, clusters, prec_par = 5)
```

Arguments

`C`	`NxN` matrix with a non-negligible number of zeros. For instance, a similarity matrix estimated from a `NxD` data matrix whose rows are statistical units, or a co-association matrix resulting from clustering ensembles.
`clusters`	A vector of integers from `1:k` (with `k <= 4`) indicating a partition of the N units resulting from clustering.
`prec_par`	Optional argument. The maximum number of alternative pivots for each group.

Details

Consider H distinct partitions of a set of N d-dimensional statistical units into k groups determined by some clustering technique. A $N \times N$ co-association matrix C with generic element $c_{i,j} = n_{i,j}/H$ can be constructed, where $n_{i,j}$ is the number of times the i-th and the j-th unit are assigned to the same cluster with respect to the clustering ensemble. Units which are very distant from each other are likely to have zero co-occurrences; as a consequence, C is a square symmetric matrix expected to contain a non-negligible number of zeros. The main task of the MUS algorithm is to detect submatrices of small rank from the co-association matrix and extract those units—pivots—such that the $k \times k$ submatrix of C, determined by only the pivotal rows and columns indexes, is identical or nearly identical. Practically, the resulting units have the desirable property to be representative of the group they belong to.

Value

`pivots`	The `k` pivotal units

References

Egidi, L., Pappadà, R., Pauli, F., Torelli, N. (2018). Maxima Units Search(MUS) algorithm: methodology and applications. In: Perna, C. , Pratesi, M., Ruiz-Gazen A. (eds.) Studies in Theoretical and Applied Statistics, Springer Proceedings in Mathematics and Statistics 227, pp. 71–81.

Examples

```
# Data generated from a mixture of three bivariate Gaussian distributions

N <- 620
centers  <- 3
n1 <- 20
n2 <- 100
n3 <- 500
x  <- matrix(NA, N,2)
truegroup <- c( rep(1,n1), rep(2, n2), rep(3, n3))

for (i in 1:n1){
 x[i,]=rmvnorm(1, c(1,5), sigma=diag(2))}
for (i in 1:n2){
 x[n1+i,]=rmvnorm(1, c(4,0), sigma=diag(2))}
for (i in 1:n3){
 x[n1+n2+i,]=rmvnorm(1, c(6,6), sigma=diag(2))}

# Build a similarity matrix from clustering ensembles

H <- 1000
a <- matrix(NA, H, N)

for (h in 1:H){
   a[h,] <- kmeans(x,centers)$cluster
}

sim_matr <- matrix(1, N,N)
for (i in 1:(N-1)){
  for (j in (i+1):N){
     sim_matr[i,j] <- sum(a[,i]==a[,j])/H
     sim_matr[j,i] <- sim_matr[i,j]
     }
}

# Obtain a clustering solution via kmeans with multiple random seeds

cl <- KMeans(x, centers)$cluster

# Find three pivots

mus_alg <- MUS(C = sim_matr, clusters = cl)
```

---

piv_KMeans *k-means Clustering Using Pivotal Algorithms For Seeding*

Description

Perform classical k-means clustering on a data matrix using pivots as initial centers.

Usage

```
piv_KMeans(x, centers, alg.type = c("KMeans", "hclust"),
  piv.criterion = c("MUS", "maxsumint", "maxsumnoint", "maxsumdiff"),
  H = 1000, ...)
```

Arguments

`x`	A $N \times D$ data matrix, or an object that can be coerced to such a matrix (such as a numeric vector or a dataframe with all numeric columns).
`centers`	The number of clusters in the solution.
`alg.type`	The clustering algorithm for the initial partition of the N units into the desired number of clusters. Possible choices are `"KMeans"` and `"hclust"`.
`piv.criterion`	The pivotal criterion used for identifying one pivot for each group. Possible choices are: `"MUS"`, `"maxsumint"`, `"maxsumnoint"`,`"maxsumdiff"`. If `centers <= 4`, the default method is `"MUS"`; otherwise, the default method is `"maxsumdiff"` (see the details and the vignette).
`H`	If `"MUS"` is selected, this is the number of distinct k-means partitions used for building a $N \times N$ co-association matrix.
`...`	Optional arguments to be passed to `MUS` or `KMeans`.

Details

The function implements a modified version of k-means which aims at improving the clustering solution starting from a careful seeding. In particular, it performs a pivot-based initialization step using pivotal methods to find the initial centers for the clustering procedure. The starting point consists of multiple runs of the classical k-means (which uses random seeds via `Kmeans` function of the `RcmdrMisc` package) with a fixed number of clusters in order to build the co-association matrix of data units.

Value

A list with components

`cluster`	A vector of integers indicating the cluster to which each point is allocated.
`centers`	A matrix of cluster centres (centroids).
`totss`	The total sum of squares.
`withinss`	The within-cluster sum of squares for each cluster.
`tot.withinss`	The within-cluster sum of squares summed across clusters.
`betwennss`	The between-cluster sum of squared distances.
`size`	The number of points in each cluster.
`iter`	The number of (outer) iterations.
`ifault`	integer: indicator of a possible algorithm problem – for experts.
`pivots`	The pivotal units identified by the selected pivotal criterion.

Author(s)

Leonardo Egidi legidi@units.it

References

Egidi, L., Pappadà, R., Pauli, F., Torelli, N. (2018). K-means seeding via MUS algorithm. Conference Paper, Book of Short Papers, SIS2018, ISBN: 9788891910233.

Examples

```
# Data generated from a mixture of three bivariate Gaussian distributions

N  <- 620
k  <- 3
n1 <- 20
n2 <- 100
n3 <- 500
x  <- matrix(NA, N,2)
truegroup <- c( rep(1,n1), rep(2, n2), rep(3, n3))

for (i in 1:n1){
 x[i,]=rmvnorm(1, c(1,5), sigma=diag(2))}
for (i in 1:n2){
 x[n1+i,]=rmvnorm(1, c(4,0), sigma=diag(2))}
for (i in 1:n3){
 x[n1+n2+i,]=rmvnorm(1, c(6,6), sigma=diag(2))}

# Apply piv_KMeans with MUS as pivotal criterion

res <- piv_KMeans(x, k)

# Apply piv_KMeans with maxsumdiff as pivotal criterion

res2 <- piv_KMeans(x, k, piv.criterion ="maxsumdiff")

# Plot the data and the clustering solution

par(mfrow=c(1,2), pty="s")
colors_cluster <- c("grey", "darkolivegreen3", "coral")
colors_centers <- c("black", "darkgreen", "firebrick")
plot(x, col = colors_cluster[truegroup],
   bg= colors_cluster[truegroup], pch=21, xlab="x[,1]",
   ylab="x[,2]", cex.lab=1.5,
   main="True data", cex.main=1.5)

plot(x, col = colors_cluster[res$cluster],
   bg=colors_cluster[res$cluster], pch=21, xlab="x[,1]",
   ylab="x[,2]", cex.lab=1.5,
   main="piv_KMeans", cex.main=1.5)
points(x[res$pivots[1],1], x[res$pivots[1],2],
   pch=24, col=colors_centers[1],bg=colors_centers[1],
   cex=1.5)
points(x[res$pivots[2],1], x[res$pivots[2],2],
   pch=24,  col=colors_centers[2], bg=colors_centers[2],
   cex=1.5)
```

```
points(x[res$pivots[3],1], x[res$pivots[3],2],
   pch=24, col=colors_centers[3], bg=colors_centers[3],
   cex=1.5)
points(res$centers, col = colors_centers[1:k],
   pch = 8, cex = 2)
```

`piv_MCMC` *JAGS Sampling for Gaussian Mixture Models and Clustering via Co-Association Matrix.*

Description

Perform MCMC JAGS sampling for Gaussian mixture models, post-process the chains and apply a clustering technique to the MCMC sample. Pivotal units for each group are selected among four alternative criteria.

Usage

```
piv_MCMC(y, k, priors, nMC, piv.criterion = c("MUS", "maxsumint",
  "maxsumnoint", "maxsumdiff"), clustering = c("diana", "hclust"))
```

Arguments

`y`	N-dimensional data vector/matrix.
`k`	Number of mixture components.
`priors`	Input prior hyperparameters (see Details).
`nMC`	Number of MCMC iterations for the JAGS function execution.
`piv.criterion`	The pivotal criterion used for identifying one pivot for each group. Possible choices are: `"MUS"`, `"maxsumint"`, `"maxsumnoint"`,`"maxsumdiff"`. If `k <= 4`, the default method is `"MUS"`; otherwise, the default method is `"maxsumdiff"` (see the Details and the vignette).
`clustering`	The clustering technique adopted for partitioning the `N` observations into `k` groups. Possible choices: `"diana"` (default), `"hclust"`.

Details

The function fits univariate and bivariate Bayesian Gaussian mixture models of the form (here for univariate only):

$$(Y_i|Z_i = j) \sim \mathcal{N}(\mu_j, \phi_j),$$

where the Z_i, $i = 1, \dots, N$, are i.i.d. random variables, $j = 1, \dots, k$, ϕ_j is the group variance, $Z_i \in 1, \dots, k$ are the latent group allocation, and

$$P(Z_i = j) = \pi_j.$$

The likelihood of the model is then

$$L(y; \mu, \pi, \phi) = \prod_{i=1}^{N} \sum_{j=1}^{k} \pi_j \mathcal{N}(\mu_j, \phi_j),$$

where $(\mu, \phi) = (\mu_1, \dots, \mu_k, \phi_1, \dots, \phi_k)$ are the component-specific parameters and $\pi = (\pi_1, \dots, \pi_k)$ the mixture weights. Let ν denote a permutation of $1, \dots, k$, and let $\nu(\mu) = (\mu_{\nu(1)}, \dots, \mu_{\nu(k)})$,

$\nu(\phi) = (\phi_{\nu(1)}, \ldots, \phi_{\nu(k)})$, $\nu(\pi) = (\pi_{\nu(1)}, \ldots, \pi_{\nu(k)})$ be the corresponding permutations of μ, ϕ and π. Denote by V the set of all the permutations of the indexes $1, \ldots, k$, the likelihood above is invariant under any permutation $\nu \in V$, that is

$$L(y; \mu, \pi, \phi) = L(y; \nu(\mu), \nu(\pi), \nu(\phi)).$$

As a consequence, the model is unidentified with respect to an arbitrary permutation of the labels. When Bayesian inference for the model is performed, if the prior distribution $p_0(\mu, \pi, \phi)$ is invariant under a permutation of the indices, then so is the posterior. That is, if $p_0(\mu, \pi, \phi) = p_0(\nu(\mu), \nu(\pi), \phi)$, then

$$p(\mu, \pi, \phi | y) \propto p_0(\mu, \pi, \phi) L(y; \mu, \pi, \phi)$$

is multimodal with (at least) $k!$ modes.

Priors are chosen as weakly informative. For univariate mixtures, the specification is the same as the function `BMMmodel` of the `bayesmix` package:

$$\mu_j \sim \mathcal{N}(0, 1/B0inv)$$

$$\phi_j \sim \text{invGamma}(nu0Half, nu0S0Half)$$

$$\pi \sim \text{Dirichlet}(1, \ldots, 1)$$

$$S0 \sim \text{Gamma}(g0Half, g0G0Half),$$

with default values: $B0inv = 0.1, nu0Half = 10, S0 = 2, nu0S0Half = nu0Half \times S0, g0Half = 5e{-}17, g0G0Half = 5e{-}33$, in accordance with the default specification: \ `priors=list(kind = "independence", p`
\ (see `bayesmix` for further details and choices).

For bivariate mixtures, the prior specification is the following:

$$\boldsymbol{\mu}_j \sim \mathcal{N}_2(\boldsymbol{\mu}_0, S2)$$

$$1/\Sigma \sim \text{Wishart}(S3, 3)$$

$$\pi \sim \text{Dirichlet}(1, \ldots, 1),$$

where $S2$ and $S3$ are diagonal matrices with diagonal elements (the variances) equal to 1e+05. The user may specify other values for the hyperparameters $\boldsymbol{\mu}_0, S2, S3$ via `priors` argument in such a way:\

`priors =list(mu0 = c(1,1), S2 = ...,S3 = ...),\`

with the constraint for $S2$ and $S3$ to be positive definite.

The function performs JAGS sampling using the `bayesmix` package for univariate Gaussian mixtures, and the `runjags` package for bivariate Gaussian mixtures. After MCMC sampling, this function clusters the units in `k` groups, calls the `piv_sel()` function and yields the pivots obtained from one among four different methods (the user may specify one among them via `piv.criterion` argument): `"maxsumint"`, `"maxsumnoint"`, `"maxsumdiff"` and `"MUS"` (available only if `k <= 4`) (see the vignette for thorough details).

Value

The function gives the MCMC output, the clustering solutions and the pivotal indexes. Here is a complete list of outputs.

`Freq` — `k x 2` matrix where: the first column reports the number of units allocated to each group as given by JAGS program; the second column reports the same number of units as given by the chains' post-processing.

true.iter	The number of MCMC iterations for which the number of JAGS groups exactly coincides with the prespecified number of groups k.
z	N x k x true.iter array with values: 1, if the i-th unit belongs to the j-th group at the h-th iteration; 0, otherwise.
ris	MCMC output matrix as provided by JAGS.
groupPost	true.iter x N matrix with values from 1:k indicating the post-processed group allocation vector.
mu_switch	If y is a vector, a true.iter x k matrix with the post-processed MCMC chains for the mean parameters; if y is a matrix, a true.iter x 2 x k array with the post-processed MCMC chains for the mean parameters.
mu_raw	If y is a vector, a nMC x k matrix with the raw MCMC chains for the mean parameters as given by JAGS; if y is a matrix, a nMC x 2 x k array with the raw MCMC chains for the mean parameters as given by JAGS.
C	Co-association matrix constructed from the MCMC sample.
grr	Group vector allocation as provided by "diana" or "hclust".
pivots	The pivotal units identified by the selected pivotal criterion.
piv.criterion	Gives the pivotal criterion used for identifying the pivots.

Author(s)

Leonardo Egidi legidi@units.it

References

Egidi, L., Pappadà, R., Pauli, F. and Torelli, N. (2018). Relabelling in Bayesian Mixture Models by Pivotal Units. Statistics and Computing, 28(4), 957-969.

Examples

```
# Bivariate simulation

N   <- 200
k   <- 4
nMC <- 1000
M1  <-c(-.5,8)
M2  <- c(25.5,.1)
M3  <- c(49.5,8)
M4  <- c(63.0,.1)
Mu  <- matrix(rbind(M1,M2,M3,M4),c(4,2))
stdev    <- cbind(rep(1,k), rep(200,k))
Sigma.p1 <- matrix(c(stdev[1,1],0,0,stdev[1,1]), nrow=2, ncol=2)
Sigma.p2 <- matrix(c(stdev[1,2],0,0,stdev[1,2]), nrow=2, ncol=2)
W <- c(0.2,0.8)
sim <- piv_sim(N,k,Mu, stdev, Sigma.p1,Sigma.p2,W)
res <- piv_MCMC(y = sim$y, k =k, nMC = nMC)
#changing priors
res2 <- piv_MCMC(y = sim$y,
                 priors = list (
                 mu0=c(1,1),
                 S2 = matrix(c(0.002,0,0, 0.1),2,2, byrow=TRUE),
                 S3 = matrix(c(0.1,0,0,0.1), 2,2, byrow =TRUE)),
```

```
                  k = k, nMC = nMC)


# Fishery data (bayesmix package)

data(fish)
y <- fish[,1]
k <- 5
nMC <- 5000
res <- piv_MCMC(y = y, k = k, nMC = nMC)
# changing priors
res2   <- piv_MCMC(y = y,
                   priors = list(kind = "condconjugate",
                   parameter = "priorsRaftery",
                   hierarchical = "tau"),  k =k, nMC = nMC)
```

---

piv_plot *Plotting outputs from pivotal relabelling*

Description

Plot and visualize MCMC outputs, posterior relabelled chains and estimates and diagnostics.

Usage

```
piv_plot(y, mcmc, rel_est, type = c("chains", "estimates", "hist"))
```

Arguments

y	Data vector or matrix.
mcmc	The ouptut of the raw MCMC sampling, as provided by piv_MCMC.
rel_est	Pivotal estimates as provided by piv_rel.
type	Type of plots required. Choose among: "chains", "estimates", "hist".

Author(s)

Leonardo Egidi legidi@units.it

Examples

```
# Fishery data

data(fish)
y <- fish[,1]
N <- length(y)
k <- 5
nMC <- 5000
res <- piv_MCMC(y = y, k = k, nMC = nMC)
rel <- piv_rel(mcmc=res, nMC = nMC)
piv_plot(y, res, rel, "chains")
```

```
piv_plot(y, res, rel, "estimates")
piv_plot(y, res, rel, "hist")
```

---

piv_rel *Perfroming the pivotal relabelling step and computing the relabelled posterior estimates*

Description

This function allows to perform the pivotal relabelling procedure described in Egidi et al. (2018) and to obtain the relabelled posterior estimates.

Usage

```
piv_rel(mcmc, nMC)
```

Arguments

mcmc	The output of the MCMC sampling from `piv_MCMC`.
nMC	The number of total MCMC iterations (given in input to the `piv_MCMC` function, or any function suited for MCMC sampling).

Details

Prototypical models in which the label switching problem arises are mixture models, as explained in the Details section of the `piv_MCMC` function.

These models are unidentified with respect to an arbitrary permutation of the labels $1, ..., k$. Relabelling means permuting the labels at each iteration of the Markov chain in such a way that the relabelled chain can be used to draw inferences on component-specific parameters.

We assume here that an MCMC sample is obtained from the posterior distribution for model above–for instance via `piv_MCMC` function–with a prior distribution which is labelling invariant. Furthermore, suppose that we can find k units, one for each group, which are (pairwise) separated with (posterior) probability one (that is, the posterior probability of any two of them being in the same group is zero). It is then straightforward to use the k units, called pivots in what follows, to identify the groups and to relabel the chains: for each MCMC iteration $h = 1, \dots, H$ (H corresponds to the argument `nMC`) and group $j = 1, \dots, k$, set

$$[\mu_j]_h = [\mu_{[Z_{i_j}]_h}]_h;$$

$$[Z_i]_h = j \text{ for } i : [Z_i]_h = [Z_{i_j}]_h.$$

The applicability of this strategy is limited by the existence of the pivots, which is not guaranteed. The existence of the pivots is a requirement of the method, meaning that its use is restricted to those chains—or those parts of a chain—for which the pivots are present. First, although the model is based on a mixture of k components, each iteration of the chain may imply a different number of non-empty groups. Let then $[k]_h \leq k$ be the number of non-empty groups at iteration h,

$$[k]_h = \#\{j : [Z_i]_h = j \text{ for some } i\},$$

where $\#A$ is the cardinality of the set A. Hence, the relabelling procedure outlined above can be used only for the subset of the chain for which $[k]_h = k$; let it be

$$\mathcal{H}_k = \{h : [k]_h = k\},$$

which correspond to the argument `true.iter` given by `piv_MCMC`. This means that the resulting relabelled chain is not a sample (of size H) from the posterior distribution, but a sample (of size $\#\mathcal{H}_k$) from the posterior distribution conditional on there being (exactly) k non-empty groups. Even if k non-empty groups are available, however, there may not be k perfectly separated units. Let us define

$$\mathcal{H}^*_k = \{h \in \mathcal{H}_k : \exists k, s \text{ s.t. } [Z_{i_k}]_h = [Z_{i_s}]_h\}$$

that is, the set of iterations where (at least) two pivots are in the same group. In order for the pivot method to be applicable, we need to exclude iterations $\mathcal{H}^*_k$; that is, we can perform the pivot relabelling on $\| - \mathcal{H}^*_k$, corresponding to the argument `Final_It`.

Value

This function gives the relabelled posterior estimates–both mean and medians–obtained from the Markov chains of the MCMC sampling.

`mu_rel_mean` k-vector (in case of univariate mixture) or `k x 2` matrix (in case of bivariate mixture) of estimated posterior means for the mean parameters.

`mu_rel_median` k-vector (in case of univariate mixture) or `k x 2` matrix (in case of bivariate mixture) of estimated posterior medians for the mean parameters.

`mu_rel_complete`
Complete relabelled chains

`Final_It` The final number of valid MCMC iterations, as explained in Details

Author(s)

Leonardo Egidi legidi@units.it

References

Egidi, L., Pappada, R., Pauli, F. and Torelli, N. (2018). Relabelling in Bayesian Mixture Models by Pivotal Units. Statistics and Computing, 28(4), 957-969.

Examples

```
#Univariate simulation

N   <- 250
nMC <- 2500
k   <- 3
p   <- rep(1/k,k)
x   <- 3
stdev <- cbind(rep(1,k), rep(200,k))
Mu    <- seq(-trunc(k/2)*x,trunc(k/2)*x,length=k)
W     <- c(0.2,0.8)
sim   <- piv_sim(N,k,Mu,stdev,W=W)
res   <- piv_MCMC(y = sim$y, k =k, nMC = nMC)
rel   <- piv_rel(mcmc=res, nMC = nMC)
```

```
#Bivariate simulation

N <- 200
k <- 3
nMC <- 5000
M1  <- c(-.5,8)
M2  <- c(25.5,.1)
M3  <- c(49.5,8)
Mu  <- matrix(rbind(M1,M2,M3),c(k,2))
stdev <- cbind(rep(1,k), rep(200,k))
Sigma.p1 <- matrix(c(stdev[1,1],0,0,stdev[1,1]),
                   nrow=2, ncol=2)
Sigma.p2 <- matrix(c(stdev[1,2],0,0,stdev[1,2]),
                   nrow=2, ncol=2)
W <- c(0.2,0.8)
sim <- piv_sim(N,k,Mu,stdev,Sigma.p1,Sigma.p2,W)
res <- piv_MCMC(y = sim$y, k = k, nMC = nMC)
rel <- piv_rel(mcmc = res, nMC = nMC)
piv_plot(y=sim$y, mcmc=res, rel_est = rel, type="chains")
piv_plot(y=sim$y, mcmc=res, rel_est = rel,
         type="hist")
```

---

piv_sel *Pivotal Selection via Co-Association Matrix*

Description

Finding the pivots according to three different methods involving a co-association matrix C.

Usage

```
piv_sel(C, clusters)
```

Arguments

C	A $N \times N$ co-association matrix, i.e. a matrix whose elements are co-occurences of pair of units in the same cluster among H distinct partitions.
clusters	A vector of integers indicating a partition of the N units into, say, k groups.

Details

Given a set of N observations $(y_1, y_2, ..., y_N)$ (y_i may be a d-dimensional vector, $d \geq 1$, consider clustering methods to obtain H distinct partitions into k groups. The matrix C is the co-association matrix, where $c_{i,p} = n_{i,p}/H$, with $n_{i,p}$ the number of times the pair (y_i, y_p) is assigned to the same cluster among the H partitions.

Let j be the group containing units $\mathcal{J}_j$, the user may choose $i^* \in \mathcal{J}_j$ that maximizes one of the quantities:

$$\sum_{p \in \mathcal{J}_j} c_{i^* p}$$

or

$$\sum_{p \in \mathcal{J}_j} c_{i^*p} - \sum_{j \notin \mathcal{J}_j} c_{i^*p}.$$

These methods give the unit that maximizes the global within similarity (`"maxsumint"` and the unit that maximizes the difference between global within and between similarities `"maxsumdiff"`, respectively. Alternatively, we may choose $i^* \in \mathcal{J}_j$, which minimizes:

$$\sum_{p \notin \mathcal{J}_j} c_{i^*p},$$

obtaining the most distant unit among the members that minimize the global dissimilarity between one group and all the others (`"maxsumnoint"`). See the vignette for further details.

Value

`pivots` A matrix with k rows and three columns containing the indexes of the pivotal units for each method.

Author(s)

Leonardo Egidi legidi@units.it

References

Egidi, L., Pappadà, R., Pauli, F. and Torelli, N. (2018). Relabelling in Bayesian Mixture Models by Pivotal Units. Statistics and Computing, 28(4), 957-969.

Examples

```
# Iris data

data(iris)
x<- iris[,1:4]
N <- length(iris[,1])
H <- 1000
a <- matrix(NA, H, N)

# Perform H k-means partitions

for (h in 1:H){
 a[h,] <- kmeans(x, centers = 3)$cluster
}
# Build the co-association matrix

C <- matrix(1, N,N)
for (i in 1:(N-1)){
 for (j in (i+1):N){
   C[i,j] <- sum(a[,i]==a[,j])/H
   C[j,i] <- C[i,j]
 }}

km <- kmeans(x, centers =3)

# Find the pivots according to the three possible pivoyal criterion
```

```
ris <- piv_sel(C, clusters = km$cluster)

plot(iris[,1], iris[,2], xlab ="Sepal.Length", ylab= "Sepal.Width",
col = km$cluster)

 # Add the pivots according to maxsumdiff criterion

points( x[ris$pivot[,3], c( "Sepal.Length","Sepal.Width" )], col = 1:3,
cex =2, pch = 8 )
```

`piv_sim` *Generate Data from a Gaussian Nested Mixture*

Description

Simulate N observations from a nested Gaussian mixture model with k pre-specified components under uniform group probabilities $1/k$, where each group is in turn drawn from a further level consisting of two subgroups.

Usage

```
piv_sim(N, k, Mu, stdev, Sigma.p1 = matrix(c(1, 0, 0, 1), 2, 2, byrow =
  TRUE), Sigma.p2 = matrix(c(100, 0, 0, 100), 2, 2, byrow = TRUE),
  W = c(0.5, 0.5))
```

Arguments

N	The desired sample size.
k	The desired number of mixture components.
Mu	The input mean vector/matrix.
stdev	A k x2 matrix of input standard deviations, one for each group (from 1 to k) and for each subgroup (from 1 to 2). For univariate mixtures only.
Sigma.p1	The covariance matrix for the first subgroup. For bivariate mixtures only.
Sigma.p2	The covariance matrix for the second subgroup. For bivariate mixtures only.
W	The vector for the mixture weights of the two subgroups,

Details

The functions allows to simulate values from a double (nested) univariate Gaussian mixture:

$$(Y_i|Z_i = j) \sim \sum_{s=1}^{2} p_{js}\,\mathcal{N}(\mu_j, \sigma^2_{js}),$$

or from a bivariate nested Gaussian mixture:

$$(Y_i|Z_i = j) \sim \sum_{s=1}^{2} p_{js}\,\mathcal{N}_2(\boldsymbol{\mu}_j, \Sigma_s),$$

where σ_{js}^2 is the variance for the group j and the subgroup s (stdev is the argument for specifying the $kx2$ standard deviations for univariate mixtures); Σ_s is the covariance matrix for the subgroup $s, s = 1, 2$, where the two matrices are specified by Sigma.p1 and Sigma.p2 respectively; μ_j and $\boldsymbol{\mu}_j,\ j = 1, \ldots, k$ are the mean input vector and matrix respectively, specified by the argument Mu; W is a vector of dimension 2 for the subgroups weights.

Value

y	The N simulated observations.
true.group	A vector of integers from 1:k indicating the values of the latent variables Z_i.
subgroups	A 2 x N matrix with values 1 or 2 indicating the subgroup to which each observation is drawn from.

Examples

```
# Bivariate mixture simulation with three components

N  <- 2000
k  <- 3
M1 <- c(-45,8)
M2 <- c(45,.1)
M3 <- c(100,8)
Mu <- matrix(rbind(M1,M2,M3),c(k,2))
stdev    <- cbind(rep(1,k), rep(200,k))
Sigma.p1 <- matrix(c(stdev[1,1],0,0,stdev[1,1]),
nrow=2, ncol=2)
Sigma.p2 <- matrix(c(stdev[1,2],0,0,stdev[1,2]),
 nrow=2, ncol=2)
W   <- c(0.2,0.8)
sim <- piv_sim(N, k, Mu, Sigma.p1 = Sigma.p1,
Sigma.p2 = Sigma.p2, W)
plot(sim$y, xlab="y[,1]", ylab="y[,2]")
```

Index