Package 'pivmet'

October 14, 2022

Type Package

Title Pivotal Methods for Bayesian Relabelling and k-Means Clustering

Version 0.4.0

Date 2021-04-28

Author Leonardo Egidi[aut, cre],
Roberta Pappadà[aut],
Francesco Pauli[aut],
Nicola Torelli[aut]

Maintainer Leonardo Egidi <legidi@units.it>

License GPL-2

Description Collection of pivotal algorithms
for: relabelling the MCMC chains in order to undo the label
switching problem in Bayesian mixture models,
as proposed in Egidi, Pappadà, Pauli and Torelli (2018a)<doi:10.1007/s11222-017-9774-2>;
initializing the centers of the classical k-means algorithm
in order to obtain a better clustering solution. For further details see
Egidi, Pappadà, Pauli and Torelli (2018b)<ISBN:9788891910233>.

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

Encoding UTF-8

SystemRequirements pandoc (>= 1.12.3), pandoc-citeproc

Depends R (>= 3.1.0)

Imports cluster, mclust, MASS, corpcor, runjags, rstan, bayesmix,
rjags, mvtnorm, bayesplot

Suggests knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 7.1.1

BuildManual yes

NeedsCompilation no

Repository CRAN

Date/Publication 2021-04-30 10:50:02 UTC

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 = 10)
```

Arguments

`C`	$N \times N$ matrix with a non-negligible number of zeros. For instance, a similarity matrix estimated from a $N \times D$ data matrix whose rows are statistical units, or a co-association matrix resulting from clustering ensembles.
`clusters`	A vector of integers from $1 : k$ indicating the cluster to which each point is allocated (it requires $k < 5$, see Details).
`prec_par`	Optional argument. The maximum number of candidate pivots for each group. Default is 10.

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.

With the argument `prec_par` the user may increase the powerful of the underlying MUS algorithm (see @egidi2018mus for details). Given the default value 10, the function internally computes an

effective prec_par as $\min(10, \min n_j)$, where n_j is the number of units belonging to the group j, $j = 1, \ldots, k$.

Value

pivots	A vector of integers in 1:N denoting the indeces of the k selcted pivotal units.
prec_par	The effective number of alternative pivots considered for each group. See Details.

Author(s)

Leonardo Egidi <legidi@units.it>, Roberta Pappadà

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

## Not run:
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))


 x[1:n1,]=rmvnorm(n1, c(1,5), sigma=diag(2))
 x[(n1+1):(n1+n2),]=rmvnorm(n2, c(4,0), sigma=diag(2))
 x[(n1+n2+1):(n1+n2+n3),]=rmvnorm(n3, 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(NA, 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)

## End(Not run)
```

---

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"),
  method = "average",
  piv.criterion = c("MUS", "maxsumint", "minsumnoint", "maxsumdiff"),
  H = 1000,
  iter.max = 10,
  nstart = 10,
  prec_par = 10
)
```

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 groups for the the k-means solution.
alg.type	The clustering algorithm for the initial partition of the N units into the desired number of clusters. Possible choices are "kmeans" (default) and "hclust".
method	If alg.type is "hclust", the character string defining the clustering method. The methods implemented are "single", "complete", "average", "ward.D", "ward.D2", "mcquitty", "median", "centroid". The default is "average".

piv.criterion	The pivotal criterion used for identifying one pivot for each group. Possible choices are: "MUS", "maxsumint", "minsumnoint","maxsumdiff". If centers <= 4, the default method is "MUS"; otherwise, the default method is "maxsumint" (see the details and the vignette).
H	The number of distinct k-means runs used for building the $N \times N$ co-association matrix. Default is 10^3.
iter.max	If alg.type is "kmeans", the maximum number of iterations to be passed to kmeans(). Default is 10.
nstart	If alg.type is "kmeans", the number of different starting random seeds to be passed to kmeans(). Default is 10.
prec_par	If piv.criterion is "MUS", the maximum number of competing pivots in each group. If groups' sizes are less than the default value, which is 10, then it is set equal to the cardinality of the smallest group in the initial partition.

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 by selecting nstart>1 in the kmeans function, 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 centers (centroids).
coass	The co-association matrix built from ensemble clustering.
pivots	The pivotal units identified by the selected pivotal criterion.
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).

Author(s)

Leonardo Egidi <legidi@units.it>, Roberta Pappada

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

## Not run:
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))

 x[1:n1,] <- rmvnorm(n1, c(1,5), sigma=diag(2))
 x[(n1+1):(n1+n2),] <- rmvnorm(n2, c(4,0), sigma=diag(2))
 x[(n1+n2+1):(n1+n2+n3),] <- rmvnorm(n3, 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")
graphics::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)

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

## End(Not run)
```

---

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

Description

Perform MCMC JAGS sampling or HMC Stan 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,
  nMC,
  priors,
  piv.criterion = c("MUS", "maxsumint", "minsumnoint", "maxsumdiff"),
  clustering = c("diana", "hclust"),
  software = c("rjags", "rstan"),
  burn = 0.5 * nMC,
  chains = 4,
  cores = 1
)
```

Arguments

y	N-dimensional vector for univariate data or $N \times D$ matrix for multivariate data.
k	Number of mixture components.
nMC	Number of MCMC iterations for the JAGS/Stan function execution.
priors	Input prior hyperparameters (see Details for default options).
piv.criterion	The pivotal criterion used for identifying one pivot for each group. Possible choices are: "MUS", "maxsumint", "minsumnoint","maxsumdiff". The default method is "maxsumint" (see the Details and the vignette).
clustering	The algorithm adopted for partitioning the N observations into k groups. Possible choices are "diana" (default) or "hclust" for divisive and agglomerative hierarchical clustering, respectively.
software	The selected MCMC method to fit the model: "rjags" for the JAGS method, "rstan" for the Stan method. Default is "rjags".
burn	The burn-in period (only if method "rjags" is selected). Default is 0.5$\times$ nMC.
chains	A positive integer specifying the number of Markov chains. The default is 4.
cores	The number of cores to use when executing the Markov chains in parallel (only if software="rstan"). Default is 1.

Details

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

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

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

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

The likelihood of the model is then

$$L(y; \mu, \eta, \sigma) = \prod_{i=1}^{N} \sum_{j=1}^{k} \eta_j \mathcal{N}(\mu_j, \sigma_j),$$

where $(\mu, \sigma) = (\mu_1, \ldots, \mu_k, \sigma_1, \ldots, \sigma_k)$ are the component-specific parameters and $\eta = (\eta_1, \ldots, \eta_k)$ the mixture weights. Let ν denote a permutation of $1, \ldots, k$, and let $\nu(\mu) = (\mu_{\nu(1)}, \ldots, \mu_{\nu(k)})$, $\nu(\sigma) = (\sigma_{\nu(1)}, \ldots, \sigma_{\nu(k)})$, $\nu(\eta) = (\eta_{\nu(1)}, \ldots, \eta_{\nu(k)})$ be the corresponding permutations of μ, σ and η. 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

$$L(y; \mu, \eta, \sigma) = L(y; \nu(\mu), \nu(\eta), \nu(\sigma)).$$

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 $p_0(\mu, \eta, \sigma)$ is invariant under a permutation of the indices, then so is the posterior. That is, if $p_0(\mu, \eta, \sigma) = p_0(\nu(\mu), \nu(\eta), \sigma)$, then

$$p(\mu, \eta, \sigma | y) \propto p_0(\mu, \eta, \sigma) L(y; \mu, \eta, \sigma)$$

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

Depending on the selected software, the model parametrization changes in terms of the prior choices. Precisely, the JAGS philosophy with the underlying Gibbs sampling is to use noninformative priors, and conjugate priors are preferred for computational speed. Conversely, Stan adopts weakly informative priors, with no need to explicitly use the conjugacy. For univariate mixtures, when `software="rjags"` the specification is the same as the function `BMMmodel` of the `bayesmix` package:

$$\mu_j \sim \mathcal{N}(\mu_0, 1/B0inv)$$

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

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

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

with default values: $\mu_0 = 0, B0inv = 0.1, nu0Half = 10, S0 = 2, nu0S0Half = nu0Half \times S0, g0Half = 5e-17, g0G0Half = 5e-33$, in accordance with the default specification:

`priors=list(kind = "independence", parameter = "priorsFish", hierarchical = "tau")`

(see `bayesmix` for further details and choices).

When `software="rstan"`, the prior specification is:

$$\mu_j \sim \mathcal{N}(\mu_0, 1/B0inv)$$

$$\sigma_j \sim \text{Lognormal}(\mu_\sigma, \tau_\sigma)$$

$$\eta_j \sim \text{Uniform}(0,1),$$

with default values: $\mu_0 = 0, B0inv = 0.1, \mu_\sigma = 0, \tau_\sigma = 2$. The users may specify new hyperparameter values with the argument:

`priors=list(mu_0=1, B0inv=0.2, mu_sigma=3, tau_sigma=5)`

For multivariate mixtures, when `software="rjags"` the prior specification is the following:

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

$$\Sigma^{-1} \sim \text{Wishart}(S3, D+1)$$

$$\eta \sim \text{Dirichlet}(\boldsymbol{\alpha}),$$

where $\boldsymbol{\alpha}$ is a k-dimensional vector and S_2 and S_3 are positive definite matrices. By default, $\boldsymbol{\mu}_0 = \mathbf{0}$, $\boldsymbol{\alpha} = (1, \ldots, 1)$ and S_2 and S_3 are diagonal matrices, with diagonal elements equal to 1e+05. The user may specify other values for the hyperparameters $\boldsymbol{\mu}_0, S_2, S_3$ and $\boldsymbol{\alpha}$ via `priors` argument in such a way:

`priors =list(mu_0 = c(1,1), S2 = ..., S3 = ..., alpha = ...)`

with the constraint for $S2$ and $S3$ to be positive definite, and $\boldsymbol{\alpha}$ a vector of dimension k with nonnegative elements.

When `software="rstan"`, the prior specification is:

$$\boldsymbol{\mu}_j \sim \mathcal{N}_D(\boldsymbol{\mu}_0, LD * L^T)$$

$$L \sim \text{LKJ}(\epsilon)$$

$$D_j^* \sim \text{HalfCauchy}(0, \sigma_d).$$

The covariance matrix is expressed in terms of the LDL decomposition as $LD * L^T$, a variant of the classical Cholesky decomposition, where L is a $D \times D$ lower unit triangular matrix and $D*$ is a $D \times D$ diagonal matrix. The Cholesky correlation factor L is assigned a LKJ prior with ϵ degrees of freedom, which, combined with priors on the standard deviations of each component, induces a prior on the covariance matrix; as $\epsilon \to \infty$ the magnitude of correlations between components decreases, whereas $\epsilon = 1$ leads to a uniform prior distribution for L. By default, the hyperparameters are $\boldsymbol{\mu}_0 = \mathbf{0}$, $\sigma_d = 2.5, \epsilon = 1$. The user may propose some different values with the argument:

`priors=list(mu_0=c(1,2), sigma_d = 4, epsilon =2)`

If `software="rjags"` the function performs JAGS sampling using the `bayesmix` package for univariate Gaussian mixtures, and the `runjags` package for multivariate Gaussian mixtures. If `software="rstan"` the function performs Hamiltonian Monte Carlo (HMC) sampling via the `rstan` package (see the vignette and the Stan project for any help).

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"`, `"minsumnoint"`, `"maxsumdiff"` and `"MUS"` (available only if `k <= 4`) (see the vignette for thorough details). Due to computational reasons clarified in the Details section of the function `piv_rel`, the length of the MCMC chains will be minor or equal than the input argument `nMC`; this length, corresponding to the value `true.iter` returned by the procedure, is the number of MCMC iterations for which the number of JAGS/Stan groups exactly coincides with the prespecified number of groups `k`.

Value

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

`true.iter`	The number of MCMC iterations for which the number of JAGS/Stan groups exactly coincides with the prespecified number of groups `k`.
`Mu`	An estimate of the groups' means.
`groupPost`	$true.iter \times N$ matrix with values from `1:k` indicating the post-processed group allocation vector.
`mcmc_mean`	If `y` is a vector, a $true.iter \times k$ matrix with the post-processed MCMC chains for the mean parameters; if `y` is a matrix, a $true.iter \times D \times k$ array with the post-processed MCMC chains for the mean parameters.
`mcmc_sd`	If `y` is a vector, a $true.iter \times k$ matrix with the post-processed MCMC chains for the sd parameters; if `y` is a matrix, a $true.iter \times D$ array with the post-processed MCMC chains for the sd parameters.
`mcmc_weight`	A $true.iter \times k$ matrix with the post-processed MCMC chains for the weights parameters.
`mcmc_mean_raw`	If `y` is a vector, a $(nMC - burn) \times k$ matrix with the raw MCMC chains for the mean parameters as given by JAGS; if `y` is a matrix, a $(nMC - burn) \times D \times k$ array with the raw MCMC chains for the mean parameters as given by JAGS/Stan.
`mcmc_sd_raw`	If `y` is a vector, a $(nMC - burn) \times k$ matrix with the raw MCMC chains for the sd parameters as given by JAGS/Stan; if `y` is a matrix, a $(nMC - burn) \times D$ array with the raw MCMC chains for the sd parameters as given by JAGS/Stan.
`mcmc_weight_raw`	A $(nMC - burn) \times k$ matrix with the raw MCMC chains for the weights parameters as given by JAGS/Stan.
`C`	The $N \times N$ co-association matrix constructed from the MCMC sample.
`grr`	The vector of cluster membership returned by `"diana"` or `"hclust"`.
`pivots`	The vector of indices of pivotal units identified by the selected pivotal criterion.
`model`	The JAGS/Stan model code. Apply the `"cat"` function for a nice visualization of the code.
`stanfit`	An object of S4 class `stanfit` for the fitted model (only if `software="rstan"`).

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

## Not run:
N   <- 200
k   <- 4
D   <- 2
nMC <- 1000
M1  <- c(-.5,8)
M2  <- c(25.5,.1)
M3  <- c(49.5,8)
M4  <- c(63.0,.1)
Mu  <- rbind(M1,M2,M3,M4)
Sigma.p1 <- diag(D)
Sigma.p2 <- 20*diag(D)
W <- c(0.2,0.8)
sim <- piv_sim(N = N, k = k, Mu = Mu,
               Sigma.p1 = Sigma.p1,
               Sigma.p2 = Sigma.p2, W = W)

## rjags (default)
res <- piv_MCMC(y = sim$y, k =k, nMC = nMC)

## rstan
res_stan <- piv_MCMC(y = sim$y, k =k, nMC = nMC,
                     software ="rstan")

# changing priors
res2 <- piv_MCMC(y = sim$y,
                 priors = list (
                 mu_0=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)

## End(Not run)


### Fishery data (bayesmix package)

## Not run:
library(bayesmix)
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)

## End(Not run)
```

---

piv_plot *Plotting outputs from pivotal relabelling*

Description

Plot and visualize MCMC outputs and posterior relabelled chains/estimates.

Usage

```
piv_plot(
  y,
  mcmc,
  rel_est,
  par = c("mean", "sd", "weight", "all"),
  type = c("chains", "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.
par	The parameters for which estimates are displayed. Choose among: "mean", "sd", "weight" and "all".
type	Type of plots required. Choose among: "chains", "hist".

Author(s)

Leonardo Egidi <legidi@units.it>

Examples

```
# Fishery data
## Not run:
library(bayesmix)
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")

## End(Not run)
```

---

piv_rel *Performing 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)
```

Arguments

mcmc	The output of the MCMC sampling from piv_MCMC.

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 a MCMC sample is obtained for the posterior distribution of a Gaussian mixture model–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 and denoted by the indexes $i_1, \ldots, i_k$, to identify the groups and to relabel the chains: for each MCMC iteration $h = 1, \ldots, H$ (H corresponds to the argument nMC) and group $j = 1, \ldots, k$, 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

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

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 $\mathcal{H}_k - \mathcal{H}_k^*$, 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.

`final_it`	The final number of valid MCMC iterations, as explained in Details.
`final_it_p`	The proportion of final valid MCMC iterations.
`rel_mean`	The relabelled chains of the means: a `final_it` $\times k$ matrix for univariate data, or a `final_it` $\times D \times k$ array for multivariate data.
`rel_sd`	The relabelled chains of the sd's: a `final_it` $\times k$ matrix for univariate data, or a `final_it` $\times D$ matrix for multivariate data.
`rel_weight`	The relabelled chains of the weights: a `final_it` $\times k$ matrix.

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

```
#Univariate simulation
## Not run:
N   <- 250
nMC <- 2500
```

```
k   <- 3
p   <- rep(1/k,k)
x   <- 3
stdev <- cbind(rep(1,k), rep(20,k))
Mu    <- seq(-trunc(k/2)*x,trunc(k/2)*x,length=k)
W     <- c(0.2,0.8)
sim   <- piv_sim(N = N, k = k, Mu = Mu,
                 stdev = stdev, W=W)
res   <- piv_MCMC(y = sim$y, k =k, nMC = nMC)
rel   <- piv_rel(mcmc=res)

## End(Not run)

#Bivariate simulation
## Not run:
N <- 200
k <- 3
D <- 2
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))
Sigma.p1 <- diag(D)
Sigma.p2 <- 20*diag(D)
W <- c(0.2,0.8)
sim <- piv_sim(N = N, k = k, Mu = Mu,
               Sigma.p1 = Sigma.p1,
               Sigma.p2 = Sigma.p2, W = W)
res <- piv_MCMC(y = sim$y, k = k, nMC = nMC)
rel <- piv_rel(mcmc = res)
piv_plot(y=sim$y, mcmc=res, rel_est = rel, type="chains")
piv_plot(y=sim$y, mcmc=res, rel_est = rel,
         type="hist")

## End(Not run)
```

`piv_sel` *Pivotal Selection via Co-Association Matrix*

Description

Finding pivotal units from a data partition and a co-association matrix C according to three different methods.

Usage

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

Arguments

C	A $N \times N$ co-association matrix, i.e. a matrix whose elements are co-occurrences of pair of units in the same cluster among H distinct partitions.
clusters	A vector of integers from $1:k$ 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 ("minsumnoint"). 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)
# select the columns of variables
x<- iris[,1:4]
N <- nrow(x)
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(NA, 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)

# Apply three pivotal criteria to the co-association matrix

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

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

 # Add the pivots chosen by the maxsumdiff criterion

points( x[ris$pivots[,3], 1:2], 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 = diag(2),
  Sigma.p2 = 100 * diag(2),
  W = c(0.5, 0.5)
)
```

Arguments

N	The desired sample size.
k	The desired number of mixture components.
Mu	The input mean vector of length k for univariate Gaussian mixtures; the input $k \times D$ matrix with the means' coordinates for multivariate Gaussian mixtures.
stdev	For univariate mixtures, the $k \times 2$ matrix of input standard deviations, where the first column contains the parameters for subgroup 1, and the second column contains the parameters for subgroup 2.
Sigma.p1	The $D \times D$ covariance matrix for the first subgroup. For multivariate mixtures only.
Sigma.p2	The $D \times D$ covariance matrix for the second subgroup. For multivariate 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:

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

or from a multivariate nested Gaussian mixture:

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

where σ_{js}^2 is the variance for the group j and the subgroup s (stdev is the argument for specifying the k x 2 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 $k \times N$ matrix where each row contains the index subgroup for the observations in the k-th group.

Examples

```
# Bivariate mixture simulation with three components

N  <- 2000
k  <- 3
D <- 2
M1 <- c(-45,8)
M2 <- c(45,.1)
M3 <- c(100,8)
Mu <- rbind(M1,M2,M3)
Sigma.p1 <- diag(D)
Sigma.p2 <- 20*diag(D)
W   <- c(0.2,0.8)
sim <- piv_sim(N = N, k = k, Mu = Mu, Sigma.p1 = Sigma.p1,
Sigma.p2 = Sigma.p2, W = W)
graphics::plot(sim$y, xlab="y[,1]", ylab="y[,2]")
```

Index