Package 'pmclust'

February 11, 2021

Version 0.2-1

Date 2021-02-08

Title Parallel Model-Based Clustering using Expectation-Gathering-Maximization Algorithm for Finite Mixture Gaussian Model

Depends R (>= 3.0.0), pbdMPI (>= 0.4-2)

Imports methods, MASS

Enhances MixSim

LazyLoad yes

LazyData yes

Description Aims to utilize model-based clustering (unsupervised) for high dimensional and ultra large data, especially in a distributed manner. The code employs 'pbdMPI' to perform a expectation-gathering-maximization algorithm for finite mixture Gaussian models. The unstructured dispersion matrices are assumed in the Gaussian models. The implementation is default in the single program multiple data programming model. The code can be executed through 'pbdMPI' and MPI' implementations such as 'OpenMPI' and 'MPICH'.
See the High Performance Statistical Computing website <https://snoweye.github.io/hpsc/> for more information, documents and examples.

License GPL (>= 2)

URL https://pbdr.org/

BugReports https://github.com/snoweye/pmclust/issues

MailingList Please send questions and comments to wccsnow@gmail.com

NeedsCompilation yes

Maintainer Wei-Chen Chen <wccsnow@gmail.com>

Author Wei-Chen Chen [aut, cre], George Ostrouchov [aut]

Repository CRAN

Date/Publication 2021-02-11 14:50:06 UTC

R topics documented:

`pmclust-package` *Parallel Model-Based Clustering*

Description

The pmclust aims to utilize model-based clustering (unsupervised) for high dimensional and ultra large data, especially in a distributed manner. The package employs pbdMPI to perform a parallel version of expectation and maximization (EM) algorithm for finite mixture Gaussian models. The unstructured dispersion matrices are assumed in the Gaussian models. The implementation is default in the single program multiple data (SPMD) programming model. The code can be executed through pbdMPI and independent to most MPI applications. See the High Performance Statistical Computing (HPSC) website for more information, documents and examples.

Details

Package:	pmclust
Type:	Package
License:	GPL
LazyLoad:	yes

The main function is pmclust implementing the parallel EM algorithm for mixture multivariate Gaussian models with unstructured dispersions. This function groups a data matrix X.gbd or X.spmd into K clusters where X.gbd or X.spmd is potentially huge and taken from the global environment .GlobalEnv or .pmclustEnv.

Other main functions em.step, aecm.step, apecm.step, and apecma.step may provide better performance than the em.step in terms of computing time and convergent iterations.

kmeans.step provides the fastest clustering among above algorithms, but it is restricted by Euclidean distance and spherical dispersions.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov

References

Programming with Big Data in R Website: https://pbdr.org/

Chen, W.-C. and Maitra, R. (2011) "Model-based clustering of regression time series data via APECM – an AECM algorithm sung to an even faster beat", *Statistical Analysis and Data Mining*, **4**, 567-578.

Chen, W.-C., Ostrouchov, G., Pugmire, D., Prabhat, M., and Wehner, M. (2013) "A Parallel EM Algorithm for Model-Based Clustering with Application to Explore Large Spatio-Temporal Data", *Technometrics*, (revision).

Dempster, A.P., Laird, N.M. and Rubin, D.B. (1977) "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society Series B*, **39**, 1-38.

Lloyd., S. P. (1982) "Least squares quantization in PCM", *IEEE Transactions on Information Theory*, **28**, 129-137.

Meng, X.-L. and Van Dyk, D. (1997) "The EM Algorithm – an Old Folk-song Sung to a Fast New Tune", *Journal of the Royal Statistical Society Series B*, **59**, 511-567.

See Also

em.step, aecm.step, apecm.step,
apecma.step, kmeans.step.

Examples

```
## Not run:
### Under command mode, run the demo with 2 processors by
### (Use Rscript.exe for windows system)
mpiexec -np 2 Rscript -e 'demo(gbd_em,"pmclust",ask=F,echo=F)'
mpiexec -np 2 Rscript -e 'demo(gbd_aecm,"pmclust",ask=F,echo=F)'
mpiexec -np 2 Rscript -e 'demo(gbd_apecm,"pmclust",ask=F,echo=F)'
mpiexec -np 2 Rscript -e 'demo(gbd_apecma,"pmclust",ask=F,echo=F)'
mpiexec -np 2 Rscript -e 'demo(gbd_kmeans,"pmclust",ask=F,echo=F)'

mpiexec -np 2 Rscript -e 'demo(ex_em,"pmclust",ask=F,echo=F)'
mpiexec -np 2 Rscript -e 'demo(ex_aecm,"pmclust",ask=F,echo=F)'
```

```
mpiexec -np 2 Rscript -e 'demo(ex_apecm,"pmclust",ask=F,echo=F)'
mpiexec -np 2 Rscript -e 'demo(ex_apecma,"pmclust",ask=F,echo=F)'
mpiexec -np 2 Rscript -e 'demo(ex_kmeans,"pmclust",ask=F,echo=F)'

## End(Not run)
```

`assign.N.sample` *Obtain a Set of Random Samples for X.spmd*

Description

This utility function samples data randomly from `X.spmd` to form a relatively small subset of original data. The EM algorithm on the smaller subset is topically performing fast and capturing rough structures of entire dataset.

Usage

```
assign.N.sample(total.sample = 5000, N.org.spmd)
```

Arguments

`total.sample`	a total number of samples which will be selected from the original data `X.spmd`.
`N.org.spmd`	the original data size, i.e. `nrow(X.spmd)`.

Details

This utility function performs simple random sampling without replacement for the original dataset `X.spmd`. Different random seeds should be set before calling this function.

Value

A list variable will be returned and containing:

`N`	total sample size across all S processors
`N.spmd`	sample size of given processor
`N.allspmds`	a collection of sample sizes for all S processors
`ID.spmd`	index of selected samples ranged from 1 to `N.org.spmd`

Note that `N` and `N.allspmds` are the same across all S processors, but `N.spmd` and `ID.spmd` are most likely all distinct. The lengths of these elements are 1 for `N` and `N.spmd`, S for `N.allspmd`, and `N.spmd` for `ID.spmd`.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

set.global

Examples

```
## Not run:
# Save code in a file "demo.r" and run in 4 processors by
# > mpiexec -np 4 Rscript demo.r

### Setup environment.
library(pmclust, quiet = TRUE)
comm.set.seed(123)

### Generate an example data.
N.org.spmd <- 5000 + sample(1:1000, 1)
ret.spmd <- assign.N.sample(total.sample = 5000, N.org.spmd)
cat("Rank:", comm.rank(), " Size:", ret.spmd$N.spmd,
    "\n", sep = "")

### Quit.
finalize()

## End(Not run)
```

EM-like algorithms *EM-like Steps for GBD*

Description

The EM-like algorithm for model-based clustering of finite mixture Gaussian models with unstructured dispersions.

Usage

```
em.step(PARAM.org)
aecm.step(PARAM.org)
apecm.step(PARAM.org)
apecma.step(PARAM.org)
kmeans.step(PARAM.org)
```

Arguments

PARAM.org	an original set of parameters generated by set.global.

Details

A global variable called X.spmd should exist in the .pmclustEnv environment, usually the working environment. The X.spmd is the data matrix to be clustered, and this matrix has a dimension N.spmd by p.

A PARAM.org will be a local variable inside all EM-linke functions em.step, aecm.step, apecm.step, apecma.step, and kmeans.step, This variable is a list containing all parameters related to models. This function also updates in the parameters by the EM-like algorithms, and return the convergent results. The details of list elements are initially generated by set.global.

Value

A convergent results will be returned the other list variable containing all new parameters which represent the components of models. See the help page of PARAM or PARAM.org for details.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

Chen, W.-C. and Maitra, R. (2011) "Model-based clustering of regression time series data via APECM – an AECM algorithm sung to an even faster beat", *Statistical Analysis and Data Mining*, **4**, 567-578.

Chen, W.-C., Ostrouchov, G., Pugmire, D., Prabhat, M., and Wehner, M. (2013) "A Parallel EM Algorithm for Model-Based Clustering with Application to Explore Large Spatio-Temporal Data", *Technometrics*, (revision).

Dempster, A.P., Laird, N.M. and Rubin, D.B. (1977) "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of the Royal Statistical Society Series B*, **39**, 1-38.

Lloyd., S. P. (1982) "Least squares quantization in PCM", *IEEE Transactions on Information Theory*, **28**, 129-137.

Meng, X.-L. and Van Dyk, D. (1997) "The EM Algorithm.an Old Folk-song Sung to a Fast New Tune", *Journal of the Royal Statistical Society Series B*, **59**, 511-567.

See Also

set.global, mb.print.

Examples

```
## Not run:
# Save code in a file "demo.r" and run in 4 processors by
# > mpiexec -np 4 Rscript demo.r

### Setup environment.
library(pmclust, quiet = TRUE)
comm.set.seed(123)
```

```
### Generate an example data.
N.allspmds <- rep(5000, comm.size())
N.spmd <- 5000
N.K.spmd <- c(2000, 3000)
N <- 5000 * comm.size()
p <- 2
K <- 2
data.spmd <- generate.basic(N.allspmds, N.spmd, N.K.spmd, N, p, K)
X.spmd <- data.spmd$X.spmd

### Run clustering.
PARAM.org <- set.global(K = K)          # Set global storages.
# PARAM.org <- initial.em(PARAM.org)    # One initial.
PARAM.org <- initial.RndEM(PARAM.org)   # Ten initials by default.
PARAM.new <- apecma.step(PARAM.org)     # Run APECMa.
em.update.class()                       # Get classification.

### Get results.
N.CLASS <- get.N.CLASS(K)
comm.cat("# of class:", N.CLASS, "\n")

### Quit.
finalize()

## End(Not run)
```

`generate.basic` *Generate Examples for Testing*

Description

This function will generate a small set of data for testing algorithms.

Usage

```
generate.basic(N.allspmds, N.spmd, N.K.spmd, N, p, K)
```

Arguments

`N.allspmds`	a collection of sample sizes for all S processors, i.e. a vector of length S.
`N.spmd`	total sample size of given processor.
`N.K.spmd`	sample size of each clusters given processor, i.e. sum over `N.K.spmd` is `N.spmd`, a vector of length K.
`N`	total sample size across all S processors, i.e. sum over `N.spmd` is `N`.
`p`	dimension of data `X.spmd`, i.e. `ncol(X.spmd)`.
`K`	number of clusters.

Details

For all S processors, this function will generate in total N observations from K clusters in p dimensions.

The clusters centers and dispersions are generated automatically inside the code. Currently, it is not allowed for users to change, but it is not difficult to specify them by mimicking this code.

Value

A set of simulated data and information will be returned in a list variable including:

K	number of clusters, as the input
p	dimension of data X.spmd, as the input
N	total sample size, as the input
N.allspmds	a collection of sample sizes for all S processors, as the input
N.spmd	total sample size of given processor, as the input
N.K.spmd	sample size of each clusters given processor, as the input
X.spmd	generated data set with dimension with dimension N.spmd * p
CLASS.spmd	true id of each data, a vector of length N.spmd and has values from 1 to K
N.CLASS.spmd	true sample size of each clusters, a vector of length K

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

generate.MixSim.

Examples

```
## Not run:
# Examples can be found in the help pages of em.step(),
# aecm.step(), apecm.step(), and apecma.step().

## End(Not run)
```

---

generate.MixSim *Generate MixSim Examples for Testing*

Description

This function utilizes **MixSim** to generate sets of data for testing algorithms.

Usage

```
generate.MixSim(N, p, K, MixSim.obj = NULL, MaxOmega = NULL,
                BarOmega = NULL, PiLow = 1.0, sph = FALSE, hom = FALSE)
```

Arguments

N	total sample size across all S processors, i.e. sum over N.spmd is N.
p	dimension of data X.spmd, i.e. ncol(X.spmd).
K	number of clusters.
MixSim.obj	an object returned from MixSim.
MaxOmega	maximum overlap as in MixSim.
BarOmega	averaged overlap as in MixSim.
PiLow	lower bound of mixture proportion as in MixSim.
sph	sph as in MixSim.
hom	hom as in MixSim.

Details

If MixSim.obj is NULL, then BarOmega and MaxOmega will be used in MixSim to obtain a new MixSim.obj.

Value

A set of simulated data and information will be returned in a list variable including:

K	number of clusters, as the input
p	dimension of data X.spmd, as the input
N	total sample size, as the input
N.allspmds	a collection of sample sizes for all S processors, as the input
N.spmd	total sample size of given processor, as the input
X.spmd	generated data set with dimension with dimension N.spmd * p
CLASS.spmd	true id of each data, a vector of length N.spmd and has values from 1 to K
N.CLASS.spmd	true sample size of each clusters, a vector of length K
MixSim.obj	the true model where data X.spmd generated from

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Melnykov, V., Chen, W.-C. and Maitra, R. (2012) "MixSim: Simulating Data to Study Performance of Clustering Algorithms", *Journal of Statistical Software*, (accepted).

Programming with Big Data in R Website: https://pbdr.org/

See Also

generate.basic.

Examples

```
## Not run:
# Save code in a file "demo.r" and run in 4 processors by
# > mpiexec -np 4 Rscript demo.r

### Setup environment.
library(pmclust, quiet = TRUE)

### Generate an example data.
N <- 5000
p <- 2
K <- 2
data.spmd <- generate.MixSim(N, p, K, BarOmega = 0.01)
X.spmd <- data.spmd$X.spmd

### Run clustering.
PARAM.org <- set.global(K = K)          # Set global storages.
# PARAM.org <- initial.em(PARAM.org)    # One initial.
PARAM.org <- initial.RndEM(PARAM.org)   # Ten initials by default.
PARAM.new <- apecma.step(PARAM.org)     # Run APECMa.
em.update.class()                       # Get classification.

### Get results.
N.CLASS <- get.N.CLASS(K)
comm.cat("# of class:", N.CLASS, "\n")
comm.cat("# of class (true):", data.spmd$N.CLASS.spmd, "\n")

### Quit.
finalize()

## End(Not run)
```

---

get.N.CLASS *Obtain Total Elements for Every Clusters*

Description

This function will collect the total elements for every clusters from all processors that the all reduced calls with the sum operation will be performed.

The get.CLASS returns class ids.

Usage

```
get.N.CLASS(K)

get.CLASS(PARAM)
```

Arguments

K	the total number of clusters.
PARAM	a set of parameters.

Details

The final results are distributed in all processors including the total elements for each cluster. The global variable CLASS.spmd stores the identification for each observation on each processors. This function will first summary CLASS.spmd in K categories, then use the all reduce function with the sum operation to add the numbers by clusters. The COMM.RANK 0 will be used to take care the printing.

Value

K numbers will be returned that are the total elements for each cluster. Sum of these K numbers should be equal to N the total number of observations.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

em.step, aecm.step, apecm.step,
apecma.step, kmeans.step.

Examples

```
## Not run:
# Examples can be found in the help pages of em.step(),
# aecm.step(), apecm.step(), apecma.step(), and kmeans.step().

## End(Not run)
```

Independent logL *Independent Function for Log Likelihood*

Description

This function is for debugging only and for checking if the observed data log likelihood is consistent for each EM iteration.

Usage

```
indep.logL(PARAM)
```

Arguments

PARAM	a set of parameters.

Details

This function will provide an observed data log likelihood based on the current parameter PARAM. This function will take in information from global, but no global variables will be updated by this function.

This function also don't take care the numerical issues, so the return value may be inaccurate sometimes.

Value

An observed data log likelihood will be returned. This value can quickly compare with the log likelihood computed inside em.onestep. Small difference is allowed, but large difference indicates bugs of code or illness of data.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

set.global, em.onestep.

Examples

```
## Not run:
# This is a core function for em.estep()
# see the source code for details.
# Reset .pmclustEnv$CONTROL$debug to turn on this function
# automatically for each EM iteration.

## End(Not run)
```

`Initialization` *Initialization for EM-like Algorithms*

Description

These functions implement initialization of EM-like algorithms for model-based clustering based on `X.spmd`, and initialization of K-means algorithm by randomly picking samples from data based on `X.spmd`.

Usage

```
initial.RndEM(PARAM)
initial.em(PARAM, MU = NULL)
initial.center(PARAM, MU = NULL)
```

Arguments

`PARAM`	an original set of parameters generated by `set.global`.
`MU`	a center matrix with dim = $p \times K$.

Details

For `initial.RndEM`, the procedure is implemented by randomly picking `.pmclustEnv$CONTROL$RndEM.iter` starting points from data `X.spmd` and run one E-step to obtain the log likelihood. Then pick the starting point with the highest log likelihood as the best choice to pursue the MLEs in further EM iterations.

This function repeatedly run `initial.em` by `.pmclustEnv$CONTROL$RndEM.iter` random starts and pick the best initializations from the random starts.

For `initial.em`, it takes `X.spmd` from the global environment and randomly pick K of them as the centers of K groups. If `MU` is specified, then this `MU` will be the centers. The default identity dispersion in `PARAM$SIGMA` will be used. Then, one E-step will be called to obtain the log likelihood and new classification will be updated.

This function is used to implement the RndEM procedure for more elaborate initialization scheme in `initial.RndEM`. Potentially, several random starts should be tried before running EM algorithms. This can benefit in two aspects including: shorter convergent iterations and better classification results.

For `initial.center`, if `MU` is given, then the center will be assigned according.

Value

The best initial starting points `PARAM` will be returned among all random starting points. The number of random starting points is assigned by `set.global` to a list variable `CONTROL`. See the help page of `initial.em` and `set.global` for details.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

Maitra, R. (2009) "Initializing partition-optimization algorithms", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **6:1**, 114-157.

See Also

set.global, em.step, aecm.step,
apecm.step, apecma.step, kmeans.step.

Examples

```
## Not run:
# Examples can be found in the help page of em.step(),
# aecm.step(), apecm.step(), apecma.step(), and kmeans.step().

## End(Not run)
```

---

mb.print *Print Results of Model-Based Clustering*

Description

This function will print summarized messages for model-based clustering.

Usage

```
mb.print(PARAM, CHECK)
```

Arguments

PARAM	a set of convergent parameters to be printed.
CHECK	a set of checking parameters to be printed.

Details

This function will provide a quick summary from the PARAM and CHECK typically the output of clusterings when algorithms stop. The COMM.RANK 0 will be used to take care the printing.

Value

Summarized messages will print/cat on screen by default.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

em.step, aecm.step, apecm.step,
apecma.step.

Examples

```
## Not run:
# Examples can be found in the help pages of em.step(),
# aecm.step(), apecm.step(), and apecma.step().

## End(Not run)
```

One E-Step — *Compute One E-step and Log Likelihood Based on Current Parameters*

Description

This function will perform one E-step based on current parameters. This is a core function of em.onestep.

Usage

```
e.step(PARAM, update.logL = TRUE)
```

Arguments

PARAM	a set of parameters.
update.logL	TRUE for update observed data log likelihood.

Details

This function will base on the current parameter to compute the densities for all observations for all K components, and update the Z.spmd matrix. If the update.logL is true, then the log likelihood W.spmd.rowSums will be also updated before the end of this function.

Sum of W.spmd.rowSums of all processors will be the observed data log likelihood for the current iteration.

Value

Several global variables will be overwrote after this call including Z.spmd, W.spmd.rowSums, W.spmd, U.spmd, and Z.colSums.

Computing Issues

Since the clusters can be degenerated or highly flat, these cause very large positive or negative exponents in densities. The log likelihood will tend to be inaccurate (not finite). Since the mixture structures can be over fit, this also cause very tiny mixing proportions. The poster probabilities can also unstable (NaN).

These can be solved by rescaling the range of exponents carefully and adjust the scaling factor on the log values. See CONTROL for details about constrains on E- and M-steps.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

set.global, em.onestep, m.step.

Examples

```
## Not run:
# This is a core function for em.onestep()
# see the source code for details.

## End(Not run)
```

One M-Step *Compute One M-Step Based on Current Posterior Probabilities*

Description

This function will perform one M-step based on current posterior probabilities. This is a core function of em.onestep.

Usage

```
m.step(PARAM)
```

Arguments

PARAM	a set of parameters.

Details

This function will base on the current posterior probabilities Z.spmd to estimate the parameters PARAM mainly including mixing proportions ETA, centers of clusters MU, and dispersions of clusters SIGMA.

Value

Returning a new PARAM which maximizes the complete data log likelihood for the current iteration.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

set.global, em.onestep, e.step.

Examples

```
## Not run:
# This is a core function for em.onestep()
# see the source code for details.

## End(Not run)
```

One Step of EM algorithm

One EM Step for GBD

Description

One EM step only for model-based clustering of finite mixture Gaussian models with unstructured dispersions. This is a core function of em.step.

Usage

```
em.onestep(PARAM)
```

Arguments

PARAM	an original set of parameters generated by set.global.

Details

A global variable called X.spmd should exist in the .pmclustEnv environment, usually the working environment. The X.spmd is the data matrix to be clustered, and this matrix has a dimension N.spmd by p.

The PARAM will be a local variable for the current iteration inside em.onestep, and this variable is a list containing all parameters related to models. This function also updates in the parameters by the EM algorithm, and return a new PARAM for the next iteration. The details of list elements are initially generated by set.global.

Value

This function is one EM step. The global variables will be updated and a new PARAM will be returned. See the help page of PARAM or PARAM.org for details.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

set.global, e.step, m.step.

Examples

```
## Not run:
# This is a core function for em.step()
# see the source code for details.

## End(Not run)
```

---

pmclust and pkmeans *Parallel Model-Based Clustering and Parallel K-means Algorithm*

Description

Parallel Model-Based Clustering and Parallel K-means Algorithm

Usage

```
pmclust(X = NULL, K = 2, MU = NULL,
  algorithm = .PMC.CT$algorithm, RndEM.iter = .PMC.CT$RndEM.iter,
  CONTROL = .PMC.CT$CONTROL, method.own.X = .PMC.CT$method.own.X,
  rank.own.X = .pbd_env$SPMD.CT$rank.source, comm = .pbd_env$SPMD.CT$comm)

pkmeans(X = NULL, K = 2, MU = NULL,
  algorithm = c("kmeans"),
  CONTROL = .PMC.CT$CONTROL, method.own.X = .PMC.CT$method.own.X,
  rank.own.X = .pbd_env$SPMD.CT$rank.source, comm = .pbd_env$SPMD.CT$comm)
```

Arguments

X	a GBD row-major matrix.
K	number of clusters.
MU	pre-specified centers.
algorithm	types of EM algorithms.
RndEM.iter	number of Rand-EM iterations.
CONTROL	a control for algorithms, see CONTROL for details.
method.own.X	how X is distributed.
rank.own.X	who own X if method.own.X = "single".
comm	MPI communicator.

Details

These are high-level functions for several functions in **pmclust** including: data distribution, setting global environment .pmclustEnv, initializations, algorithm selection, etc.

The input X is in gbd. It will be converted in gbd row-major format and copied into .pmclustEnv for computation. By default, **pmclust** uses a GBD row-major format (gbdr). While common means that X is identical on all processors, and single means that X only exist on one processor rank.own.X.

Value

These functions return a list with class pmclust or pkmeans.

See the help page of PARAM or PARAM.org for details.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

set.global, e.step, m.step.

Examples

```
## Not run:
# Save code in a file "demo.r" and run in 4 processors by
# > mpiexec -np 4 Rscript demo.r

### Setup environment.
library(pmclust, quiet = TRUE)

### Load data
X <- as.matrix(iris[, -5])
```

```
### Distribute data
jid <- get.jid(nrow(X))
X.gbd <- X[jid,]

### Standardized
N <- allreduce(nrow(X.gbd))
p <- ncol(X.gbd)
mu <- allreduce(colSums(X.gbd / N))
X.std <- sweep(X.gbd, 2, mu, FUN = "-")
std <- sqrt(allreduce(colSums(X.std^2 / (N - 1))))
X.std <- sweep(X.std, 2, std, FUN = "/")

### Clustering
library(pmclust, quiet = TRUE)
comm.set.seed(123, diff = TRUE)

ret.mb1 <- pmclust(X.std, K = 3)
comm.print(ret.mb1)

ret.kms <- pkmeans(X.std, K = 3)
comm.print(ret.kms)

### Finish
finalize()

## End(Not run)
```

`print.object` *Functions for Printing or Summarizing Objects According to Classes*

Description

Several classes are declared in **pmclust**, and these are functions to print and summary objects.

Usage

```
## S3 method for class 'pmclust'
print(x, ...)
## S3 method for class 'pkmeans'
print(x, ...)
```

Arguments

`x`	an object with the class attributes.
`...`	other possible options.

Details

These are useful functions for summarizing.

Value

The results will cat or print on the STDOUT by default.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

pmclust, pkmeans.

Examples

```
## Not run:
library(pmclust, quiet = TRUE)

# Functions applied by directly type the names of objects.

## End(Not run)
```

Read Me First *Read Me First Function*

Description

This function print the annotations of all variables used in this package.

Usage

```
readme()
```

Details

This package is optimized in the way by pre-specifying several global variables in .pmclustEnv. These variables will be overwrote by EM algorithms. Users should use these names to access the results and utilize them with cautions.

Value

A readme message will print on screen by default and explain the global variables used in this package, including:

CHECK	convergent checking
CLASS.spmd	true id of each data, a vector of length N.spmd and has values from 1 to K
COMM.RANK	rank of current processor, obtained from comm.rank of pbdMPI
COMM.SIZE	total processors in MPI world, obtained from comm.size of pbdMPI
CONTROL	controls for EM iterations
PARAM	set or parameters
SAVE.param	(debug only) save parameters for every iterations
SAVE.iter	(debug only) save computing time for every iterations
U.spmd	temporary storage for density
W.spmd	temporary storage for eta * density
W.spmd.rowSums	temporary storage for rowSums of W.spmd
X.spmd	generated data set with dimension with dimension N.spmd * p
Z.colSums	temporary storage for rowSums of Z.spmd
Z.spmd	posterior probabilities
p.times.logtwopi	p * log(2 * pi)

Each variable may contain several elements if it is a list, some variables are used for temporary storages in order to optimize computing, and some variables are used for constant variables. These variables may be restricted, and only generated by the function set.global.

One can access these variables via the global environment .pmclustEnv such as .pmclustEnv$CONTROL.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

set.global.

Examples

```
## Not run:
readme()

## End(Not run)
```

`Set Global Variables` *Set Global Variables According to the global matrix X.gbd (X.spmd)*

Description

This function will set several sets of variables globally in the environment `.pmclustEnv` according to the global matrix `X.gbd/X.spmd`.

Usage

```
set.global.gbd(K = 2, X.gbd = NULL, PARAM = NULL,
    algorithm = c("em", "aecm", "apecm", "apecma", "kmeans"),
    RndEM.iter = 10)

set.global(K = 2, X.spmd = NULL, PARAM = NULL,
    algorithm = c("em", "aecm", "apecm", "apecma", "kmeans"),
    RndEM.iter = 10)
```

Arguments

`K`	an original set of parameters generated by `set.global`.
`X.gbd`	an input GBD matrix.
`X.spmd`	an input SPMD matrix.
`PARAM`	an original set of parameters generated by `set.global`.
`algorithm`	an original set of parameters generated by `set.global`.
`RndEM.iter`	number of RndEM iterations.

Details

WARNING: A global variable named `X.gbd/X.spmd` should be set before calling `set.global` where `X.gbd/X.spmd` is a matrix containing data with dimension `N.spmd * p`. i.e. `N.spmd` observations and `p` variables.

`X.gbd/X.spmd` is supposed to exist in `.GlobalEnv`. If not, they should be as an input object and will be copied into `.pmclustEnv` which is less efficient.

Value

A new set of `PARAM` will be returned and several global variables will be set according to the data `X.gbd/X.spmd`.

Sets of global variables are store in the default environment `.pmclustEnv`.

Use `readme` to see all global variables set by this function.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

em.step, aecm.step, apecm.step,
apecma.step.

Examples

```
## Not run:
# Examples can be found in the help pages of em.step(),
# aecm.step(), apecm.step(), apecma.step(), and kmeans.step().

## End(Not run)
```

Set of CONTROL *A Set of Controls in Model-Based Clustering.*

Description

This set of controls are used to guide all algorithms implemented in this package.

Format

A list variable contains several parameters for computing.

Details

.PMC.CT stores all default controls for pmclust and pkmeans including

algorithm	algorithms implemented
algorithm.gbd	algorithms implemented for gbd/spmd
method.own.X	how X is distributed
CONTROL	a CONTROL list as in next

The elements of CONTROL or .pmclustEnv$CONTROL are

max.iter	maximum number of iterations (1000)
abs.err	absolute error for convergence (1e-4)
rel.err	relative error for convergence (1e-6)
debug	debugging flag (0)
RndEM.iter	number of RndEM iterations (10)
exp.min	minimum exponent (log(.Machine$double.xmin))
exp.max	maximum exponent (log(.Machine$double.xmax))
U.min	minimum of diagonal of chol

`U.max`	maximum of diagonal of `chol`
`stop.at.fail`	stop iterations when fails such as NaN

These elements govern the computing including number of iterations, convergent criteria, ill conditions, and numerical issues. Some of them are machine dependent.

Currently, the `algorithm` could be `em`, `aecm`, `apecm`, `apecma`, and `kmeans` for GBD. The `method.own.X` could be `gbdr`, `common`, and `single`.

Numerical Issues

For example, `exp.min` and `exp.max` will control the range of densities function before taking logarithm. If the density values were no in the range, they would be rescaled. The scaling factor will be also recorded for post adjustment for observed data log likelihood. This will provide more accurate posterior probabilities and observed data log likelihood.

Also, `U.min` and `U.max` will control the output of `chol` when decomposing `SIGMA` in every E-steps. If the diagonal terms were out of the range, a `PARAM$U.check` would be set to `FALSE`. Only the components with `TRUE U.check` will estimate and update the dispersions in M-steps for the rest of iterations.

These problems may cause wrong posteriors and log likelihood due to the degenerate and inflated components. Usually, this is a sign of overestimate the number of components `K`, or the initialization do not provide good estimations for parameters. See `e.step` for more information about computing.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

`set.global.gbd`, and `set.global`.

Examples

```
## Not run:
# Use set.global() to generate one of this.
# X.spmd should be pre-specified before calling set.global().

## End(Not run)
```

Set of PARAM *A Set of Parameters in Model-Based Clustering.*

Description

This set of parameters are used in initialization, EM iterations, and final convergent results. All share the same structure in a list variable.

Format

A list variable contains several parameters for computing.

Details

The elements of PARAM or PARAM.org are

N	number of observations
p	dimension of each observation, total number of variables
K	number of clusters
ETA	mixing proportion
log.ETA	log of mixing proportion
MU	centers, dim = $p \times K$
SIGMA	dispersions, a list containing K elements, each element is a matrix, dim = $p \times p$
U	Choleski of SIGMA, the same size of SIGMA
U.check	checks of each elements of U, length K
logL	log likelihood
min.N.CLASS	minimum number of elements in a cluster (restrictions)

The model parameters are ETA, MU, and SIGMA, while log.ETA, U, U.check, and min.N.CLASS are only used in computing.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

set.global.

Examples

```
## Not run:
# Use set.global() to generate one of this.
# X.spmd should be pre-specified before calling set.global().
```

```
## End(Not run)
```

`Update Class of EM or Kmenas Results`

Update CLASS.spmd Based on the Final Iteration

Description

Update `CLASS.spmd` based on the final iteration of EM-like algorithms.

Usage

```
em.update.class()
kmeans.update.class()
```

Details

This function takes `Z.spmd` from the global environment `.pmclustEnv` and update `CLASS.spmd`, and provides the identification of groups for all data.

Value

`CLASS.spmd` will be updated.

Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and George Ostrouchov.

References

Programming with Big Data in R Website: https://pbdr.org/

See Also

`em.step`, `aecm.step`, `apecm.step`,
`apecma.step`, `kmeans.step`.

Examples

```
## Not run:
# Examples can be found in the help pages of em.step(),
# aecm.step(), apecm.step(), apecma.step(), and kmeans.step().

## End(Not run)
```

Index