

# Package ‘EMCluster’

April 28, 2017

**Version** 0.2-7

**Date** 2017-04-22

**Title** EM Algorithm for Model-Based Clustering of Finite Mixture  
Gaussian Distribution

**Depends** R (>= 3.0.1), MASS, Matrix

**Enhances** PPtree, RColorBrewer

**LazyLoad** yes

**LazyData** yes

**Description** EM algorithms and several efficient  
initialization methods for model-based clustering of finite  
mixture Gaussian distribution with unstructured dispersion  
in both of unsupervised and semi-supervised learning.

**License** Mozilla Public License 2.0

**BugReports** <https://github.com/snoweye/EMCluster/issues>

**URL** <https://github.com/snoweye/EMCluster>

**NeedsCompilation** yes

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

**Author** Wei-Chen Chen [aut, cre],  
Ranjan Maitra [aut],  
Volodymyr Melnykov [ctb]

**Repository** CRAN

**Date/Publication** 2017-04-28 07:23:36 UTC

## R topics documented:

EMCluster-package . . . . .	2
Assign Class . . . . .	3
Conversion . . . . .	4
Dataset . . . . .	6
EM Algorithm . . . . .	7

EM Control . . . . .	8
Information Criteria . . . . .	10
Initialization and EM . . . . .	11
Jaccard Index . . . . .	13
Likelihood Mixture Tests . . . . .	14
LMT Functions . . . . .	15
MVN . . . . .	16
Other Initializations . . . . .	17
Plot EM Results . . . . .	18
Plot Multivariate Data . . . . .	20
Plot Projection and Contour . . . . .	21
Post I Information Functions . . . . .	22
Print and Summary . . . . .	23
Projection On 2D . . . . .	24
Rand Index . . . . .	26
Recolor Classification IDs . . . . .	27
Single Step . . . . .	28

**Index** **30**

---

EMCluster-package	<i>EM Algorithm for Model-Based Clustering of Finite Mixture Gaussian Distribution</i>
-------------------	--

---

### Description

EMCluster provides EM algorithms and several efficient initialization methods for model-based clustering of finite mixture Gaussian distribution with unstructured dispersion in both of unsupervised and semi-supervised clustering.

### Details

Package:	EMCluster
Type:	Package
License:	GPL
LazyLoad:	yes

The install command is simply as

```
> R CMD INSTALL EMCluster_0.2-0.tar.gz
```

from a command mode or

```
R> install.packages("EMCluster")
```

inside an R session.

**Author(s)**

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[init.EM](#), [emcluster](#).

**Examples**

```
## Not run:
demo(allinit, 'EMCluster', ask = F, echo = F)
demo(allinit_ss, 'EMCluster', ask = F, echo = F)

## End(Not run)
```

---

Assign Class

*Assign Class Id*

---

**Description**

This function assigns cluster id to each observation in  $x$  according to the desired model `emobj` or specified parameters `pi`, `Mu`, and `LTSigma`.

**Usage**

```
assign.class(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL,
            lab = NULL, return.all = TRUE)
```

**Arguments**

<code>x</code>	the data matrix, dimension $n \times p$ .
<code>emobj</code>	the desired model which is a list mainly contains <code>pi</code> , <code>Mu</code> , and <code>LTSigma</code> , usually a returned object from <code>init.EM</code> .
<code>pi</code>	the mixing proportion, length $K$ .
<code>Mu</code>	the centers of clusters, dimension $K \times p$ .
<code>LTSigma</code>	the lower triangular matrices of dispersion, dimension $K \times p(p + 1)/2$ .
<code>lab</code>	labeled data for semi-supervised clustering, length $n$ .
<code>return.all</code>	if returning with a whole <code>emobj</code> object.

**Details**

This function are based either an input emobj or inputs pi, Mu, and LTSigma to assign class id to each observation of  $x$ .

If lab is submitted, then the observation with label id greater 0 will not be assigned new class.

**Value**

This function returns a list containing mainly two new variables: nc (length  $K$  numbers of observations in each class) and class (length  $n$  class id).

**Author(s)**

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[init.EM](#), [emcluster](#).

**Examples**

```
## Not run:
library(EMCluster, quietly = TRUE)
set.seed(1234)
x2 <- da2$da

ret <- init.EM(x2, nclass = 2)
ret.new <- assign.class(x2, ret, return.all = FALSE)
str(ret.new)

## End(Not run)
```

---

Conversion

*Convert Matrices in Different Format*

---

**Description**

These utility functions are to convert matrices in different formats.

**Usage**

```
LTSigma2variance(x)
variance2LTSigma(x)
LTSigma2var(x1, p = NULL)
var2LTSigma(x1)
class2Gamma(class)
Gamma2class(Gamma)
```

**Arguments**

x	a matrix/array to be converted, the dimension could be $K \times p(p + 1)/2$ or $p \times p \times K$ .
x1	a vector/matrix to be converted, the length and dimension could be $p(p + 1)/2$ and $p \times p$ .
p	dimension of matrix.
class	id of clusters for each observation, length $n$ .
Gamma	containing posterior probabilities if normalized, otherwise containing component densities weighted by mixing proportion, dimension $n \times K$ .

**Details**

LTSigma2variance converts LTSigma format to 3D array, and variance2LTSigma is the inversion function.

LTsigma2var converts LTsigma format to a matrix, and var2LTsigma is the inversion function. Note that LTsigma is one component of LTSigma.

class2Gamma converts id to a Gamma matrix where with probability 1 for the cluster where the observation belongs to, and Gamma2class converts posterior to cluster id where largest posterior is picked for each observation.

**Value**

A vector/matrix/array is returned.

**Author(s)**

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[init.EM](#), [emcluster](#).

**Examples**

```
## Not run:
library(EMcluster, quietly = TRUE)
x <- da2$LTSigma
class <- da2$class

y <- LTSigma2variance(x)
str(y)
y <- variance2LTSigma(y)
str(y)
sum(x != y)
```

```
Gamma <- class2Gamma(class)
class.new <- Gamma2class(Gamma)
sum(class != class.new)

## End(Not run)
```

---

Dataset

*Dataset for demonstrations*

---

### Description

There are four small datasets to test and demonstrate **EMCluster**.

### Usage

```
da1
da2
da3
myiris
```

### Format

da1, da2, da3 are in list, and myiris is in matrix.

### Details

da1 has 500 observations in two dimensions da1\$da\$x and da1\$da\$y, and they are in 10 clusters given in da1\$class.

da2 has 2,500 observations in two dimensions, too. The true parameters are given in da1\$pi, da1\$Mu, and da1\$LTSigma. There are 40 clusters given in da1\$class for this dataset.

da3 is similar to da2, but with lower overlaps between clusters.

myiris is selected from the original Iris dataset given by R.

### Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra.

### References

<http://maitra.public.iastate.edu/>

**Description**

These are core functions of **EMCluster** performing EM algorithm for model-based clustering of finite mixture multivariate Gaussian distribution with unstructured dispersion.

**Usage**

```
emcluster(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL,
          lab = NULL, EMC = .EMC, assign.class = FALSE)
shortemcluster(x, emobj = NULL, pi = NULL, Mu = NULL,
               LTSigma = NULL, maxiter = 100, eps = 1e-2)
simple.init(x, nclass = 1)
```

**Arguments**

<code>x</code>	the data matrix, dimension $n \times p$ .
<code>emobj</code>	the desired model which is a list mainly contains <code>pi</code> , <code>Mu</code> , and <code>LTSigma</code> , usually a returned object from <code>init.EM</code> .
<code>pi</code>	the mixing proportion, length $K$ .
<code>Mu</code>	the centers of clusters, dimension $K \times p$ .
<code>LTSigma</code>	the lower triangular matrices of dispersion, $K \times p(p + 1)/2$ .
<code>lab</code>	labeled data for semi-supervised clustering, length $n$ .
<code>EMC</code>	the control for the EM iterations.
<code>assign.class</code>	if assigning class id.
<code>maxiter</code>	maximum number of iterations.
<code>eps</code>	convergent tolerance.
<code>nclass</code>	the desired number of clusters, $K$ .

**Details**

The `emcluster` mainly performs EM iterations starting from the given parameters `emobj` without other initializations.

The `shortemcluster` performs short-EM iterations as described in `init.EM`.

**Value**

The `emcluster` returns an object `emobj` with class `emret` which can be used in post-process or other functions such as `e.step`, `m.step`, `assign.class`, `em.ic`, and `dmixmvn`.

The `shortemcluster` also returns an object `emobj` with class `emret` which is the best of several random initializations.

The `simple.init` utilizes `rand.EM` to obtain a simple initial.

**Author(s)**

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[init.EM](#), [e.step](#), [m.step](#), [.EMControl](#).

**Examples**

```
## Not run:
library(EMCluster, quietly = TRUE)
set.seed(1234)
x1 <- da1$da

emobj <- simple.init(x1, nclass = 10)
emobj <- shortemcluster(x1, emobj)
summary(emobj)

ret <- emcluster(x1, emobj, assign.class = TRUE)
summary(ret)

## End(Not run)
```

---

EM Control

*EM Control Generator and Controller*

---

**Description**

The `.EMControl` generates an EM control (`.EMC`) controlling the options and conditions of EM algorithms, i.e. this function generate a default template. One can either modify `.EMC` or employ this function to control EM algorithms. By default, `.EMC`, `.EMC.Rnd`, and `.EC.Rndp` are three native controllers as the **EMCluster** is loaded.

**Usage**

```
.EMControl(alpha = 0.99, short.iter = 200, short.eps = 1e-2,
  fixed.iter = 1, n.candidate = 3,
  EM.iter = 1000, EM.eps = 1e-6, exhaust.iter = 5)
.EMC
.EMC.Rnd
.EMC.Rndp
```



**Arguments**

<code>alpha</code>	only used in emgroup for "SVD" initialization.
<code>short.iter</code>	number of short-EM steps, default = 200.
<code>short.eps</code>	tolerance of short-EM steps, default = 1e-2.
<code>fixed.iter</code>	fixed iterations of EM for "RndEM" initialization, default = 1.
<code>n.candidate</code>	reserved for other initialization methods (unimplemented).
<code>EM.iter</code>	maximum number of long-EM steps, default = 1000.
<code>EM.eps</code>	tolerance of long-EM steps, default = 1e-6.
<code>exhaust.iter</code>	number of iterations for "exhaustEM" initialization, default = 5.

**Details**

`exhaust.iter` and `fixed.iter` are used to control the iterations of initialization procedures.

`short.iter` and `short.eps` are used to control the short-EM iterations.

`EM.iter` and `EM.eps` are used to control the long-EM iterations.

Moreover, `short.eps` and `EM.eps` are for checking convergence of the iterations.

**Value**

This function returns a list as `.EMC` by default.

The `.EMC.Rnd` is equal to `.EMControl(short.eps = Inf)` and usually used by the `rand.EM` method.

The `.EMC.Rndp` is equal to `.EMControl(fixed.iter = 5)` where each random initials run 5 EM iterations in the `rand.EM` method.

**Author(s)**

Wei-Chen Chen <[wccsnow@gmail.com](mailto:wccsnow@gmail.com)> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[init.EM](#), [emcluster](#).

**Examples**

```
## Not run:
library(EMcluster, quietly = TRUE)

.EMC <- .EMControl()
.EMC.Rnd <- .EMControl(short.eps = Inf)
.EMC.Rndp <- .EMControl(fixed.iter = 5)

## End(Not run)
```

## Description

These functions are tools for compute information criteria for the fitted models.

## Usage

```
em.ic(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL,
      llhdval = NULL)
em.aic(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
em.bic(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
em.clc(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
em.icl(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
em.icl.bic(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
```

## Arguments

<code>x</code>	the data matrix, dimension $n \times p$ .
<code>emobj</code>	the desired model which is a list mainly contains <code>pi</code> , <code>Mu</code> , and <code>LTSigma</code> , usually a returned object from <code>init.EM</code> .
<code>pi</code>	the mixing proportion, length $K$ .
<code>Mu</code>	the centers of clusters, dimension $K \times p$ .
<code>LTSigma</code>	the lower triangular matrices of dispersion, $K \times p(p + 1)/2$ .
<code>llhdval</code>	the total log likelihood value of <code>x</code> given <code>emobj</code> .

## Details

The `em.ic` calls all other functions to compute AIC (`em.aic`), BIC (`em.bic`), CLC (`em.clc`), ICL (`em.icl`), and ICL.BIC (`em.icl.bic`). All are useful information criteria for model selections, mainly choosing number of cluster.

## Value

`em.ic` returns a list containing all other information criteria for given the data `x` and the desired model `emobj`.

## Author(s)

Wei-Chen Chen <[wccsnow@gmail.com](mailto:wccsnow@gmail.com)> and Ranjan Maitra

## References

<http://maitra.public.iastate.edu/>

**See Also**

[init.EM.](#)

**Examples**

```
## Not run:
library(EMCluster, quietly = TRUE)
x2 <- da2$da

emobj <- list(pi = da2$pi, Mu = da2$Mu, LTSigma = da2$LTSigma)
em.ic(x2, emobj = emobj)

## End(Not run)
```

---

Initialization and EM *Initialization and EM Algorithm*

---

**Description**

These functions perform initializations (including `em.EM` and `RndEM`) followed by the EM iterations for model-based clustering of finite mixture multivariate Gaussian distribution with unstructured dispersion in both of unsupervised and semi-supervised clusterings.

**Usage**

```
init.EM(x, nclass = 1, lab = NULL, EMC = .EMC,
        stable.solution = TRUE, min.n = NULL, min.n.iter = 10,
        method = c("em.EM", "Rnd.EM"))
em.EM(x, nclass = 1, lab = NULL, EMC = .EMC,
      stable.solution = TRUE, min.n = NULL, min.n.iter = 10)
rand.EM(x, nclass = 1, lab = NULL, EMC = .EMC.Rnd,
        stable.solution = TRUE, min.n = NULL, min.n.iter = 10)
exhaust.EM(x, nclass = 1, lab = NULL,
           EMC = .EMControl(short.iter = 1, short.eps = Inf),
           method = c("em.EM", "Rnd.EM"),
           stable.solution = TRUE, min.n = NULL, min.n.iter = 10);
```

**Arguments**

<code>x</code>	the data matrix, dimension $n \times p$ .
<code>nclass</code>	the desired number of clusters, $K$ .
<code>lab</code>	labeled data for semi-supervised clustering, length $n$ .
<code>EMC</code>	the control for the EM iterations.
<code>stable.solution</code>	if returning a stable solution.
<code>min.n</code>	restriction for a stable solution, the minimum number of observations for every final clusters.

<code>min.n.iter</code>	restriction for a stable solution, the minimum number of iterations for trying a stable solution.
<code>method</code>	an initialization method.

### Details

The `init.EM` calls either `em.EM` if `method="em.EM"` or `rand.EM` if `method="Rnd.EM"`.

The `em.EM` has two steps: short-EM has loose convergent tolerance controlled by `.EMC$short.eps` and try several random initializations controlled by `.EMC$short.iter`, while long-EM starts from the best short-EM result (in terms of log likelihood) and run to convergence with a tight tolerance controlled by `.EMC$EM.eps`.

The `rand.EM` also has two steps: first randomly pick several random initializations controlled by `.EMC$short.iter`, and second starts from the best of the random result (in terms of log likelihood) and run to convergence.

The `lab` is only for the semi-supervised clustering, and it contains pre-labeled indices between 1 and  $K$  for labeled observations. Observations with index 0 is non-labeled and has to be clustered by the EM algorithm. Indices will be assigned by the results of the EM algorithm. See `demo(allinit_ss, 'EMCluster')` for details.

The `exhaust.EM` also calls the `init.EM` with different EMC and perform `exhaust.iter` times of EM algorithm with different initials. The best result is returned.

### Value

These functions return an object `emobj` with class `emret` which can be used in post-process or other functions such as `e.step`, `m.step`, `assign.class`, `em.ic`, and `dmixmvm`.

### Author(s)

Wei-Chen Chen <[wccsnow@gmail.com](mailto:wccsnow@gmail.com)> and Ranjan Maitra.

### References

<http://maitra.public.iastate.edu/>

### See Also

[emcluster](#), [.EMControl](#).

### Examples

```
## Not run:
library(EMCluster, quietly = TRUE)
set.seed(1234)
x <- da1$da

ret.em <- init.EM(x, nclass = 10, method = "em.EM")
ret.Rnd <- init.EM(x, nclass = 10, method = "Rnd.EM", EMC = .EMC.Rnd)

emobj <- simple.init(x, nclass = 10)
```

```
ret.init <- emcluster(x, emobj, assign.class = TRUE)

par(mfrow = c(2, 2))
plotem(ret.em, x)
plotem(ret.Rnd, x)
plotem(ret.init, x)

## End(Not run)
```

---

Jaccard Index

*Jaccard Index*

---

## Description

This function returns the Jaccard index for binary ids.

## Usage

```
Jaccard.Index(x, y)
```

## Arguments

x	true binary ids, 0 or 1.
y	predicted binary ids, 0 or 1.

## Details

All ids, x and y, should be either 0 (not active) or 1 (active). Any value other than 1 will be converted to 0.

## Value

Return the value of Jaccard index.

## Author(s)

Wei-Chen Chen <[wccsnow@gmail.com](mailto:wccsnow@gmail.com)> and Ranjan Maitra.

## References

<http://maitra.public.iastate.edu/>

**Examples**

```
## Not run:
library(EMcluster, quietly = TRUE)

x.id <- c(1, 1, 1, 0, 0, 0, 3, 3, 3)
y.id <- c(0, 1, 0, 1, 1, 1, 0, 1, 1)

Jaccard.Index(x.id, y.id)

## End(Not run)
```

---

Likelihood Mixture Tests

*Likelihood Mixture Tests*

---

**Description**

This function test two mixture Gaussian models with unstructured covariance matrix and different numbers of clusters.

**Usage**

```
lmt(emobj.0, emobj.a, x, tau = 0.5, n.mc.E.delta = 1000,
    n.mc.E.chi2, verbose = FALSE)
```

**Arguments**

emobj.0	a emret object for the null hypothesis.
emobj.a	a emret object for the alternative hypothesis.
x	the data matrix, dimension $n \times p$ .
tau	proportion of null and alternative hypotheses.
n.mc.E.delta	number of Monte Carlo simulations for expectation of delta (difference of logL).
n.mc.E.chi2	number of Monte Carlo simulations for expectation of chisquare statistics.
verbose	if verbose.

**Details**

This function calls several subroutines to compute information, likelihood ratio statistics, degrees of freedom, non-centrality of chi-squared distributions ...etc. Based on Monte Carlo methods to estimate parameters of likelihood mixture tests, this function return a p-value for testing  $H_0$ : emobj.0 v.s.  $H_a$ : emobj.a.

**Value**

A list of class lmt are returned.

**Author(s)**

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[init.EM.](#)

**Examples**

```
## Not run:
library(EMCluster, quietly = TRUE)
set.seed(1234)

x <- as.matrix(iris[, 1:4])
p <- ncol(x)
min.n <- p * (p + 1) / 2
.EMC$short.iter <- 200

ret.2 <- init.EM(x, nclass = 2, min.n = min.n, method = "Rnd.EM")
ret.3 <- init.EM(x, nclass = 3, min.n = min.n, method = "Rnd.EM")
ret.4 <- init.EM(x, nclass = 4, min.n = min.n, method = "Rnd.EM")

(lmt.23 <- lmt(ret.2, ret.3, x))
(lmt.34 <- lmt(ret.3, ret.4, x))
(lmt.24 <- lmt(ret.2, ret.4, x))

## End(Not run)
```

---

LMT Functions

*Likelihood Mixture Test (LMT) Functions of EMCluster*

---

**Description**

All likelihood mixture test (LMT) functions are for testing and can be utilized by advanced developers with caution.

Currently, these are only for workflows.

**Author(s)**

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**Description**

These functions are tools for compute density of (mixture) multivariate Gaussian distribution with unstructured dispersion.

**Usage**

```
dmvn(x, mu, LTsigma, log = FALSE)
dlmvn(x, mu, LTsigma, log = TRUE)
dmixmvn(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL, log = FALSE)
logL(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL)
```

**Arguments**

<code>x</code>	the data matrix, dimension $n \times p$ .
<code>mu</code>	the centers of clusters, length $p$ .
<code>LTsigma</code>	the lower triangular matrices of dispersion, length $p(p+1)/2$ .
<code>log</code>	if logarithm returned.
<code>emobj</code>	the desired model which is a list mainly contains <code>pi</code> , <code>Mu</code> , and <code>LTSigma</code> , usually a returned object from <code>init.EM</code> .
<code>pi</code>	the mixing proportion, length $K$ .
<code>Mu</code>	the centers of clusters, dimension $K \times p$ .
<code>LTSigma</code>	the lower triangular matrices of dispersion, $K \times p(p+1)/2$ .

**Details**

The `dmvn` and `dlmvn` compute density and log density of multivariate distribution.

The `dmixmvn` computes density of mixture multivariate distribution and is based either an input `emobj` or inputs `pi`, `Mu`, and `LTSigma` to assign class id to each observation of `x`.

The `logL` returns the value of the observed log likelihood function of the parameters at the current values of the parameters `pi`, `Mu`, and `LTSigma`, with the supplied data matrix `x`.

**Value**

A density value is returned.

**Author(s)**

Wei-Chen Chen <[wccsnow@gmail.com](mailto:wccsnow@gmail.com)> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>



**See Also**

[init.EM](#), [emcluster](#).

**Examples**

```
## Not run:
library(EMcluster, quietly = TRUE)
x2 <- da2$da
x3 <- da3$da

emobj2 <- list(pi = da2$pi, Mu = da2$Mu, LTSigma = da2$LTSigma)
emobj3 <- list(pi = da3$pi, Mu = da3$Mu, LTSigma = da3$LTSigma)

logL(x2, emobj = emobj2)
logL(x3, emobj = emobj3)

dmixmvn2 <- dmixmvn(x2, emobj2)
dmixmvn3 <- dmixmvn(x3, emobj3)

dlmvn(da2$da[1,], da2$Mu[1,], da2$LTSigma[1,])
log(dmvn(da2$da[1,], da2$Mu[1,], da2$LTSigma[1,]))

## End(Not run)
```

---

Other Initializations *Other Initializations*

---

**Description**

Two more functions with different initialization method.

**Usage**

```
starts.via.svd(x, nclass = 1, method = c("em", "kmeans"),
              EMC = .EMC)
emgroup(x, nclass = 1, EMC = .EMC)
```

**Arguments**

<code>x</code>	the data matrix, dimension $n \times p$ .
<code>nclass</code>	the desired number of clusters, $K$ .
<code>method</code>	method with the svd initializations.
<code>EMC</code>	the control for the EM iterations.

**Details**

The `starts.via.svd` utilizes SVD to initial parameters, and the `emgroup` runs the EM algorithm starting from the initial.

**Value**

The `starts.via.svd` returns an object with class `svd`, and the `emgroup` returns an object `emobj` with class `emret`.

**Author(s)**

Wei-Chen Chen <[wccsnow@gmail.com](mailto:wccsnow@gmail.com)> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[init.EM](#), [.EMControl](#).

**Examples**

```
## Not run:
library(EMcluster, quietly = TRUE)
set.seed(1234)
x1 <- da1$da

emobj <- emgroup(x1, nclass = 10)
summary(emobj)

ret.0 <- starts.via.svd(x1, nclass = 10, method = "kmeans")
summary(ret.0)

## End(Not run)
```

---

Plot EM Results

*Plot Two Dimensional Data with clusters*

---

**Description**

The functions plot two dimensional data for clusters.

**Usage**

```
plotem(emobj, x, main = NULL, xlab = NULL, ylab = NULL,
      ...)
plot2d(x, emobj = NULL, k = NULL, color.pch = 1,
      append.BN = TRUE, ...)
```

**Arguments**

<code>emobj</code>	the desired model which is a list mainly contains $\pi$ , $\mu$ , and $LTSigma$ , usually a returned object from <code>init.EM</code> .
<code>x</code>	the data matrix, dimension $n \times p$ .
<code>main</code>	title of plot.
<code>xlab</code>	label of x-axis.
<code>ylab</code>	label of y-axis.
<code>...</code>	other parameters to the plot.
<code>k</code>	index for symbols.
<code>color.pch</code>	color and style for symbols.
<code>append.BN</code>	if appending bivariate normal ellipsoid.

**Details**

This a simple x-y lot.

**Value**

A plot is returned.

**Author(s)**

Wei-Chen Chen <[wccsnow@gmail.com](mailto:wccsnow@gmail.com)> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[init.EM](#), [emcluster](#).

**Examples**

```
## Not run:
library(EMcluster, quietly = TRUE)
x1 <- da1$da

ret.1 <- starts.via.svd(x1, nclass = 10, method = "em")
summary(ret.1)

plotem(ret.1, x1)

## End(Not run)
```

---

Plot Multivariate Data

*Plot Multivariate Data*

---

### Description

The function plots multivariate data for clusters as the parallel coordinates plot.

### Usage

```
plotmd(x, class = NULL, xlab = "Variables", ylab = "Data", ...)
```

### Arguments

<code>x</code>	the data matrix, dimension $n \times p$ .
<code>class</code>	class id for all observations.
<code>xlab</code>	label of x-axis.
<code>ylab</code>	label of y-axis.
<code>...</code>	other parameters to the plot.

### Details

This a simplified parallel coordinate plot.

### Value

A plot is returned.

### Author(s)

Wei-Chen Chen <[wccsnow@gmail.com](mailto:wccsnow@gmail.com)> and Ranjan Maitra.

### References

<http://maitra.public.iastate.edu/>

### See Also

[init.EM](#), [emcluster](#).

**Examples**

```
## Not run:
library(EMCluster, quietly = TRUE)
set.seed(1234)

x <- myiris
ret <- em.EM(x, nclass = 5)
plotmd(x, ret$class)

## End(Not run)
```

---

 Plot Projection and Contour

*Plot Contour*


---

**Description**

The function plots multivariate data on 2D plane with contour. Typically, the contour is built via projection pursuit or SVD algorithms, such as [project.on.2d\(\)](#).

**Usage**

```
plotppcontour(da, Pi, Mu, S, class, class.true = NULL, n.grid = 128,
  angle = 0, xlab = "", ylab = "", main = "")
```

**Arguments**

da	a projected data matrix, dimension $n \times 2$ .
Pi	proportion, length $K$ .
Mu	the projected centers of cluster, dimension $K \times 2$ .
S	projected matrices of dispersion, dimension $p \times p \times K$ .
class	id of classifications, length $n$ .
class.true	true id of classifications if available, length $n$ .
n.grid	number of grid points.
angle	a rotation angle (0 to $2\pi$ ).
xlab	an option for <code>plot()</code> function.
ylab	an option for <code>plot()</code> function.
main	an option for <code>plot()</code> function.

**Details**

This function plots projection output of `project.on.2d()`.

da, Mu, and S are projected by some projection matrices obtained via SVD or projection pursuit algorithms. The projection is made on a 2D plane in the direction in which clusters of data x are most distinguishable to visualize.

**Value**

A 2D projection plot is returned.

**Note**

Only distinguishable for up to 7 clusters due to the limited color schemes.

**Author(s)**

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[project.on.2d\(\)](#).

**Examples**

```
## Not run:
library(EMCluster, quietly = TRUE)
library(MASS, quietly = TRUE)
set.seed(1234)

### Crabs.
x <- as.matrix(crabs[, 4:8])
ret <- init.EM(x, nclass = 4, min.n = 20)
ret.proj <- project.on.2d(x, ret)

### Plot.
pdf("crabs_ppcontour.pdf", height = 5, width = 5)
plotppcontour(ret.proj$da, ret.proj$Pi, ret.proj$Mu, ret.proj$S,
              ret.proj$class, angle = pi/6, main = "Crabs K = 4")
dev.off()

## End(Not run)
```

---

Post I Information Functions

*Post I Information Functions of EMCluster*

---

**Description**

All post I information functions are for computing relative quantities and can be utilized by advanced developers with caution.

Currently, these are only for workflows.

**Author(s)**

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

---

Print and Summary

*Functions for Printing or Summarizing Objects According to Classes*

---

**Description**

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

**Usage**

```
## S3 method for class 'emret'  
print(x, digits = max(4, getOption("digits") - 3), ...)  
## S3 method for class 'emret'  
summary(object, ...)  
## S3 method for class 'svd'  
summary(object, ...)
```

**Arguments**

x	an object with the class attributes.
digits	for printing out numbers.
object	an object with the class attributes.
...	other possible options.

**Details**

These are useful functions for summarizing and debugging.

**Value**

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

**Author(s)**

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[init.EM](#), [emcluster](#), [starts.via.svd](#).

**Examples**

```
## Not run:
library(EMcluster, quietly = TRUE)
x2 <- da2$da

emobj <- list(pi = da2$pi, Mu = da2$Mu, LTSigma = da2$LTSigma)
eobj <- e.step(x2, emobj = emobj)
emobj <- m.step(x2, emobj = eobj)
summary(emobj)

ret <- starts.via.svd(x2, nclass = 10, method = "kmeans")
summary(ret)

## End(Not run)
```

---

Projection On 2D

*Produce Projection on 2D*

---

**Description**

The function projects multivariate data on 2D plane which can be displayed by [plotppcontour\(\)](#) later.

**Usage**

```
project.on.2d(x, emobj = NULL, pi = NULL, Mu = NULL,
             LTSigma = NULL, class = NULL, method = c("PP", "SVD"))
```

**Arguments**

<code>x</code>	the data matrix, dimension $n \times p$ .
<code>emobj</code>	the desired model which is a list mainly contains <code>pi</code> , <code>Mu</code> , and <code>LTSigma</code> , usually a returned object from <code>init.EM</code> .
<code>pi</code>	the mixing proportion, length $K$ .
<code>Mu</code>	the centers of clusters, dimension $K \times p$ .
<code>LTSigma</code>	the lower triangular matrices of dispersion, $K \times p(p + 1)/2$ .
<code>class</code>	id of classifications, length $n$ .
<code>method</code>	either projection pursuit or singular value decomposition.

**Details**

This function produces projection outputs of `x` and `emobj`.



**Value**

A projection is returned which is a list contains

- `da` is a  $n \times 2$  projected matrix of `x`.
- `Pi` is the original proportion `emobj$pi` of length  $K$ .
- `Mu` is a  $K \times 2$  projected matrix of `emobj$Mu`.
- `S` is a  $2 \times 2 \times K$  projected array of `emobj$LSigma`.
- `class` is the original class id `emobj$class`.
- `proj.mat` is the projection matrix of dimension  $p$ .

**Author(s)**

Wei-Chen Chen <[wccsnow@gmail.com](mailto:wccsnow@gmail.com)> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[project.on.2d\(\)](#).

**Examples**

```
## Not run:
library(EMcluster, quietly = TRUE)
set.seed(1234)

### Iris.
x <- as.matrix(iris[, 1:4])
ret <- init.EM(x, nclass = 3, min.n = 30)
ret.proj <- project.on.2d(x, ret)

### Plot.
pdf("iris_ppcontour.pdf", height = 5, width = 5)
plotppcontour(ret.proj$da, ret.proj$Pi, ret.proj$Mu, ret.proj$S,
              ret.proj$class, main = "Iris K = 3")
dev.off()

## End(Not run)
```

---

Rand Index

*Rand Index and Adjusted Rand Index*

---

### Description

This function returns the Rand index and the adjusted Rand index for given true class ids and predicted class ids.

### Usage

```
RRand(trc1, prc1, lab = NULL)
```

### Arguments

trc1	true class ids.
prc1	predicted class ids.
lab	known ids for semi-supervised clustering.

### Details

All ids, trc1 and prc1, should be positive integers and started from 1 to K, and the maximums are allowed to be different.

lab used in semi-supervised clustering contains the labels which are known before clustering. It should be positive integer and started from 1 for labeled data and 0 for unlabeled data.

### Value

Return a Class `RRand` contains Rand index and adjusted Rand index.

### Author(s)

Wei-Chen Chen <wccsnow@gmail.com> and Ranjan Maitra.

### References

<http://maitra.public.iastate.edu/>

### Examples

```
## Not run:
library(EMCluster, quietly = TRUE)

true.id <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
pred.id <- c(2, 1, 2, 1, 1, 1, 2, 1, 1)
label   <- c(0, 0, 0, 0, 1, 0, 2, 0, 0)

RRand(true.id, pred.id)
RRand(true.id, pred.id, lab = label)
```

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

---

Recolor Classification IDs  
*Recolor Classification IDs*

---

## Description

These functions return new classification IDs.

## Usage

```
recolor(id.target, id.class, scatter.class = NULL, scatter.target = NULL)  
rematch(tg.id, cl.id)  
recode(id)
```

## Arguments

<code>id.target</code>	target class ids.
<code>id.class</code>	original class ids.
<code>scatter.class</code>	scatter class ids.
<code>scatter.target</code>	scatter target class ids.
<code>id</code>	class ids.
<code>tg.id</code>	target class ids.
<code>cl.id</code>	class ids.

## Details

The function `recolor` colors `id.target` in accordance with the most likely candidate in `id.class`. Note that if `scatter` is present, then the class given by 0 is represented as `scatter` and it is assumed to be the same for both classifications.

The function `rematch` returns a list as `id.trcl` and `id.prc1`. It is the heart of the `recolor` function and is usually called from `recolor`.

The function `recode` reorders classes to eliminate group ids without any members. It is assumed that the group ids are integers.

## Value

See Details.

## Author(s)

Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**Examples**

```
## Not run:
library(EMcluster, quietly = TRUE)

true.id <- c(1, 1, 1, 2, 2, 2, 3, 3, 3)
pred.id <- c(2, 1, 2, 1, 1, 1, 2, 1, 1)

recolor(pred.id, true.id)

## End(Not run)
```

---

Single Step

*Single E- and M-step*

---

**Description**

These functions are single E- and M-step of EM algorithm for model-based clustering of finite mixture multivariate Gaussian distribution with unstructured dispersion.

**Usage**

```
e.step(x, emobj = NULL, pi = NULL, Mu = NULL, LTSigma = NULL,
       norm = TRUE)
m.step(x, emobj = NULL, Gamma = NULL, assign.class = FALSE)
```

**Arguments**

<code>x</code>	the data matrix, dimension $n \times p$ .
<code>emobj</code>	the desired model which is a list mainly contains <code>pi</code> , <code>Mu</code> , and <code>LTSigma</code> , usually a returned object from <code>init.EM</code> .
<code>pi</code>	the mixing proportion, length $K$ .
<code>Mu</code>	the centers of clusters, dimension $K \times p$ .
<code>LTSigma</code>	the lower triangular matrices of dispersion, $K \times p(p + 1)/2$ .
<code>norm</code>	if returning normalized Gamma.
<code>Gamma</code>	containing posterior probabilities if normalized, otherwise containing component densities weighted by mixing proportion, dimension $n \times K$ .
<code>assign.class</code>	if assigning class id.

**Details**

These two functions are mainly used in debugging for development and post process after model fitting.

**Value**

The `e.step` returns a list contains Gamma, the posterior probabilities if `norm=TRUE`, otherwise it contains component densities. This is one E-step and Gamma is used to update `emobj` in the M-step next.

The `m.step` returns a new `emobj` according to the Gamma from the E-step above.

**Author(s)**

Wei-Chen Chen <[wccsnow@gmail.com](mailto:wccsnow@gmail.com)> and Ranjan Maitra.

**References**

<http://maitra.public.iastate.edu/>

**See Also**

[init.EM](#).

**Examples**

```
## Not run:
library(EMcluster, quietly = TRUE)
x2 <- da2$da

emobj <- list(pi = da2$pi, Mu = da2$Mu, LTSigma = da2$LTSigma)
eobj <- e.step(x2, emobj = emobj)
emobj <- m.step(x2, emobj = eobj)
emobj

## End(Not run)
```

# Index

- \*Topic **LMT**
  - LMT Functions, 15
- \*Topic **datasets**
  - Dataset, 6
- \*Topic **package**
  - EMCluster-package, 2
- \*Topic **post\_I\_information**
  - Post I Information Functions, 22
- \*Topic **programming**
  - EM Algorithm, 7
  - EM Control, 8
  - Initialization and EM, 11
  - Likelihood Mixture Tests, 14
  - Other Initializations, 17
- \*Topic **summary**
  - Jaccard Index, 13
  - Print and Summary, 23
  - Rand Index, 26
  - Recolor Classification IDs, 27
- \*Topic **tool**
  - Assign Class, 3
  - Information Criteria, 10
  - Single Step, 28
- \*Topic **utility**
  - Conversion, 4
  - MVN, 16
  - Plot EM Results, 18
  - Plot Multivariate Data, 20
  - Plot Projection and Contour, 21
  - Projection On 2D, 24
- .EMC (EM Control), 8
- .EMControl, 8, 12, 18
- .EMControl (EM Control), 8
- Assign Class, 3
- assign.class (Assign Class), 3
- bh.fdr (LMT Functions), 15
- class2Gamma (Conversion), 4
- col.ppcontour (LMT Functions), 15
- Conversion, 4
- da1 (Dataset), 6
- da2 (Dataset), 6
- da3 (Dataset), 6
- Dataset, 6
- d1mvn (MVN), 16
- dmixmvn (MVN), 16
- dmvn (MVN), 16
- e.step, 8
- e.step (Single Step), 28
- EM Algorithm, 7
- EM Control, 8
- em.aic (Information Criteria), 10
- em.bic (Information Criteria), 10
- em.clc (Information Criteria), 10
- em.EM (Initialization and EM), 11
- em.ic (Information Criteria), 10
- em.icl (Information Criteria), 10
- EMCluster (EMCluster-package), 2
- emcluster, 3–5, 9, 12, 17, 19, 20, 24
- emcluster (EM Algorithm), 7
- EMCluster-package, 2
- emgroup (Other Initializations), 17
- emobj object (EM Algorithm), 7
- emret class (EM Algorithm), 7
- exhaust.EM (Initialization and EM), 11
- fill.ppcontour (LMT Functions), 15
- fill.ppmu (LMT Functions), 15
- fill.pppoints (LMT Functions), 15
- Gamma2class (Conversion), 4
- GenDataSet (LMT Functions), 15
- GenMixDataSet (LMT Functions), 15
- get.cov.logit.PI (Post I Information Functions), 22
- get.cov.logit.z (Post I Information Functions), 22

- get.cov.param (Post I Information Functions), 22
- get.cov.post.z (Post I Information Functions), 22
- get.E.chi2 (LMT Functions), 15
- get.E.delta (LMT Functions), 15
- get.logor.stat (Post I Information Functions), 22
- Gmat (LMT Functions), 15
- gridOne (LMT Functions), 15
- Information Criteria, 10
- init.EM, 3–5, 8, 9, 11, 15, 17–20, 24, 29
- init.EM (Initialization and EM), 11
- Initialization and EM, 11
- Iy (LMT Functions), 15
- Iy2 (LMT Functions), 15
- Jaccard Index, 13
- Jaccard.Index (Jaccard Index), 13
- Likelihood Mixture Tests, 14
- lmt (Likelihood Mixture Tests), 14
- LMT Functions, 15
- logL (MVN), 16
- LTsigma2var (Conversion), 4
- LTsigma2variance (Conversion), 4
- m.step, 8
- m.step (Single Step), 28
- MVN, 16
- myiris (Dataset), 6
- Other Initializations, 17
- partial.f.mu.s (Post I Information Functions), 22
- partial.logit.p (Post I Information Functions), 22
- partial.logL (LMT Functions), 15
- partial.post.z (Post I Information Functions), 22
- partial.q (LMT Functions), 15
- pchisq.my (LMT Functions), 15
- Plot EM Results, 18
- Plot Multivariate Data, 20
- Plot Projection and Contour, 21
- plot2d (Plot EM Results), 18
- plotem (Plot EM Results), 18
- plotmd (Plot Multivariate Data), 20
- plotp (LMT Functions), 15
- plotppcontour, 24
- plotppcontour (Plot Projection and Contour), 21
- plotq (LMT Functions), 15
- Post I Information Functions, 22
- postPI (LMT Functions), 15
- Print and Summary, 23
- print.emret (Print and Summary), 23
- print.lmt (LMT Functions), 15
- project.on.2d, 21, 22, 25
- project.on.2d (Projection On 2D), 24
- Projection On 2D, 24
- q.map.new (LMT Functions), 15
- Rand Index, 26
- rand.EM (Initialization and EM), 11
- recode (Recolor Classification IDs), 27
- recolor (Recolor Classification IDs), 27
- Recolor Classification IDs, 27
- rematch (Recolor Classification IDs), 27
- RRand (Rand Index), 26
- shortemcluster (EM Algorithm), 7
- simple.init (EM Algorithm), 7
- Single Step, 28
- starts.via.svd, 24
- starts.via.svd (Other Initializations), 17
- summary.emret (Print and Summary), 23
- summary.svd (Print and Summary), 23
- var2LTsigma (Conversion), 4
- variance2LTsigma (Conversion), 4
- w.2 (LMT Functions), 15