

Package ‘JCM’

October 7, 2014

Title The EM Algorithm and Mixture Models

Description Fit multivariate mixture models via the EM Algorithm. Multivariate distributions include Normal distribution, t-distribution, Skew Normal distribution and and Skew t-distribution. The emmix-JCM is an updated version of EMMIX with new features such as clustering the degenerated data and fitting skew mixture models.

Depends R (>= 2.10), mvtnorm, geneplotter, feature

Version 1.0.20

Author Geoff McLachlan, Kui Wang, Angus Ng and David Peel. EMMIX was originally written in Fortran by David Peel(1999).

Maintainer Geoff McLachlan <g.mclachlan@uq.edu.au>

License file LICENCE

LazyLoad yes

Archs i386, x64

R topics documented:

bootstrap	2
conplot	4
ddmix	5
ddmsn	7
ddmst	8
ddmvn	9
ddmvt	10
do.features	11
do.jcm	12
do.kld	13
do.mix	14
do.mould	15
do.template	18
do.unzip	19
emmix	20
emmix.contours	23
emmixfit	25
emmixMOD	27

entropy	28
error.rate	29
getcov	30
getICL	31
initEmmix	32
inverse	34
jcamst	35
jcamst.predict	37
Lympho	39
msmv	39
mvt.dof	41
plotmixt.skew	42
rdemmix	45

Index	48
--------------	-----------

bootstrap

Bootstrap

Description

The standard error analysis and the bootstrap analysis of -2log(Lambda).

Usage

```
bootstrap(x,n,p,g,distr,ncov,popPAR,B=99,replace=TRUE,
          itmax=1000,epsilon=1e-5)
bootstrap.noc(x,n,p,g1,g2,distr,ncov,B=99,replace=TRUE,
              itmax=1000,epsilon=1e-5)
```

Arguments

n	The number of observations
p	The dimension of data
B	The number of simulated data or replacements to be tried
x	The dataset, an n by p numeric matrix, where n is number of observations and p the dimension of data.
g	The number of components of the mixture model
g1,g2	The range of the number of components of the mixture model
distr	A three letter string indicating the type of distribution to be fit. See Details.
ncov	A small integer indicating the type of covariance structure. See Details.
popPAR	A list with components pro, a numeric vector of the mixing proportion of each component; mu, a p by g matrix with each column as its corresponding mean; sigma, a three dimensional p by p by g array with its jth component matrix (p,p,j) as the covariance matrix for jth component of mixture models; dof, a vector of degrees of freedom for each component; delta, a p by g matrix with its columns corresponding to skew parameter vectors.
replace	A logical value indicating whether replacement to be used
itmax	A big integer specifying the maximum number of iterations to apply
epsilon	A small number used to stop the EM algorithm loop when the relative difference between log-likelihood at each iteration become sufficient small.

Details

The distribution type, `distr`, is one of the following values: "mvn" for a multivariate normal, "mvt" for a multivariate t-distribution, "msn" for a multivariate skew normal distribution and "mst" for a multivariate skew t-distribution.

The covariance matrix type, represented by the `ncov` parameter, may be any one of the following: `ncov=1` for a common variance, `ncov=2` for a common diagonal variance, `ncov=3` for a general variance, `ncov=4` for a diagonal variance, `ncov=5` for $\sigma(h)^*I(p)$ (diagonal covariance with same identical diagonal element values).

When `replace` is `FALSE`, parametric bootstrap is used; otherwise replacement method is used.

Value

`bootstrap` gives standard errors. `bootstrap.noc` returns a list with components `ret`, a B by $(g_2 - g_1)$ matrix of $-2\log(\Lambda)$, `v1k`, the loglikelihood for each g in the range of g_1 to g_2 , and `pvalue`, the p-values of g vs $g+1$. The results of fitting mixture models are stored in current working directory, which can be used via command in R: `obj <- dget("ReturnOf_g_???.ret")`.

References

McLachlan G.J. and Krishnan T. (2008). The EM Algorithm and Extensions (2nd). New Jersey: Wiley.

McLachlan G.J. and Peel D. (2000). Finite Mixture Models. New York: Wiley.

See Also

[emmmix](#), [rdemmmix](#)

Examples

```
n1=300;n2=300;n3=400;
nn <-c(n1,n2,n3)
n <- sum(nn)
p <- 2
g <- 3

sigma<-array(0,c(p,p,g))
for(h in 1:3) sigma[,,h]<-diag(p)

mu <- cbind(c(4,-4),c(3.5,4),c( 0, 0))

# for other distributions,
#delta <- cbind(c(3,3),c(1,5),c(-3,1))
#dof    <- c(3,5,5)

distr="mvn"
ncov=3

#first we generate a data set
set.seed(111) #random seed is set
dat <- rdemmmix(nn,p,g,distr,mu,sigma,dof=NULL,delta=NULL)

#start from initial partition
```

```

clust<- rep(1:g,nn)
obj <- emmixfit1(dat,g,clust,distr,ncov,itmax=1000,epsilon=1e-5)

# do bootstrap (standard error analysis)

## Not run:
std <- bootstrap(dat,n,p,g,distr,ncov,obj,B=19,
replace=TRUE,itmax=1000,epsilon=1e-5)
print(std)

# do bootstrap analysis of -2log(Lambda).

# alternatively data can be input as follow,
# dat <- read.table("mydata.txt",header=TRUE)
# p   <- ncol(dat)
# n   <- nrow(dat)

lad <- bootstrap.noc(dat,n,p,2,4,distr,ncov,B=19,
replace=FALSE,itmax=1000,epsilon=1e-5)
print(lad)

# return of g=2
obj2 <- dget("ReturnOf_g_2.ret")

# return of g=3
obj3 <- dget("ReturnOf_g_3.ret")

# return of g=4
obj4 <- dget("ReturnOf_g_4.ret")

#The posterior probability matrix for (g=3) is obtained by

tau <- obj3$tau

## End(Not run)

```

conplot

Contours

Description

These functions are called by *emmix.contours*,*emmix.filter* and *emmix.flow* to plot the contours of (skew) mixture density after fitting to the data.

Usage

```

conplot(x, y, pro, mu, sigma, dof, delta, distr, grid = 300,
nrand = 6000, levels = seq(5, 95, by = 20), col = "white")
conplot2(x, y, pro, mu, sigma, dof, delta, distr, grid = 300,
nrand = 6000, levels = seq(5, 95, by = 20))
conplot3(x, y, pro, mu, sigma, dof, delta, modpts,distr, grid =300,
nrand = 10000, levels = seq(5, 95, by = 20))

```

Arguments

x	A vector of observations on variable x.
y	A vector of observations on variable y.
pro	A vector of mixing proportions in the (skew) mixture model.
mu	A matrix with each column corresponding to the mean or location vector of one mixture component.
sigma	An array of covariance matrices for each component of the mixture distribution.
dof	A vector of degrees of freedom when "distribution" is "mvn" or "mst".
delta	A matrix with each column as skew parameter vector of one component when "distribution" is "msn" or "mst".
distr	A three letter string indicating component distribution, "mvn"=normal distribution, "mvt"=t-distribution,"msn"=skew normal distribution, "mst"=skew t-distribution.
modpts	The mode points.
grid	An integer for the number of grid points in one direction.
nrand	A large integer for the number of random numbers being drawn.
levels	A vector of contour percentage levels for the plots. It should be in the range of 0 to 100.
col	The colour of contour lines.

Details

In most case, users do not call this function directly, instead they call the function emmix.flow.

See Also

[emmix.flow](#)

ddmix

Density Functions of Mixture Models

Description

Calculate the density of multivariate mixture models at data points for each component

Usage

```
ddmix( dat, n, p, g, distr, mu, sigma, dof=NULL, delta=NULL)
```

Arguments

dat	The dataset
n	The total number of points
p	Dimension of data
g	The number of clusters
distr	A three letter string indicating the distribution; "mvn" for normal, "mvt" for t distribution, "msn" for skew normal, and "mst" for skew t distribution.

<i>mu</i>	A numeric mean matrix with each column corresponding to the mean
<i>sigma</i>	An array of dimension (p,p,g) with first two dimensions corresponding covariance matrix of each component
<i>dof</i>	A vector of degrees of freedom for each component
<i>delta</i>	A matrix with each column as skew parameter vector

Value

ddmix gives an n by g matrix of logarithm of density at each data point for each component.

References

- McLachlan G.J. and Krishnan T. (2008). The EM Algorithm and Extensions (2nd). New Jersey: Wiley.
 McLachlan G.J. and Peel D. (2000). Finite Mixture Models. New York: Wiley.

See Also

[ddmvn](#), [ddmvnt](#), [ddmsn](#), [ddmst](#).

Examples

```
p=2
g=3

#mixing proportion of each component
pro <- c(0.3,0.3,0.4)

#specify mean and covariance matrix for each component

sigma<-array(0,c(2,2,3))
for(h in 2:3) sigma[,,h]<-diag(2)
sigma[,,1]<-cbind(c(1,0),c(0,.1))

mu <- cbind(c(4,-4),c(3.5,4),c( 0, 0))

#specify other parameters for "mvt", "msn", "mst"

delta <- cbind(c(3,3),c(1,5),c(-3,1))
dof <- c(3,5,5)

#specify the distribution
distr <- "mst"

y <- c(1,2)

n=1

#then the density value at y for the mixture model is

ddmix(y, n, p, g, distr, mu, sigma, dof, delta)
```

Description

Density and random generation for Multivariate Skew Normal distributions with mean vector `mean`, covariance matrix `cov`, and skew parameter vector `del`.

Usage

```
ddmsn(dat, n, p, mean, cov, del)
rdmsn( n, p, mean, cov, del)
```

Arguments

<code>dat</code>	An n by p numeric matrix, the dataset
<code>n</code>	An integer, the number of observations
<code>p</code>	An integer, the dimension of data
<code>mean</code>	A length of p vector, the mean
<code>cov</code>	A p by p matrix, the covariance
<code>del</code>	A length of p vector, the skew parameter

Value

`ddmsn` gives the density values; `rdmsn` generates the random numbers

See Also

[rdemmix](#), [ddmvn](#), [ddmvt](#), [ddmst](#), [rdmvn](#), [rdmvt](#), [rdmst](#).

Examples

```
n <- 100
p <- 2

mean <- rep(0,p)
cov <- diag(p)
del<- c(0,1)

set.seed(3214)

y <- rdmsn( n,p,mean,cov,del)

den <- ddmsn(y,n,p,mean,cov,del)
```

ddmst

The Multivariate Skew t-distribution

Description

Density and random generation for Multivariate Skew t-distributions with mean vector `mean`, covariance matrix `cov`, degrees of freedom `nu`, and skew parameter vector `del`.

Usage

```
ddmst(dat,n, p, mean, cov, nu, del)
rdmst( n, p, mean, cov, nu, del)
```

Arguments

<code>dat</code>	An n by p numeric matrix, the dataset
<code>n</code>	An integer, the number of observations
<code>p</code>	An integer, the dimension of data
<code>mean</code>	A length of p vector, the mean
<code>cov</code>	A p by p matrix, the covariance
<code>nu</code>	A positive number, the degrees of freedom
<code>del</code>	A length of p vector, the skew parameter

Value

`ddmst` gives the density values; `rdmst` generates the random numbers

See Also

[rdemmix](#), [ddmvn](#), [ddmvt](#), [ddmsn](#), [rdmvn](#), [rdmvt](#), [rdmsn](#).

Examples

```
n <- 100
p <- 2

mean <- rep(0,p)
cov <- diag(p)
nu <- 3
del <- c(0,1)

set.seed(3214)

y <- rdmst( n,p,mean,cov,nu,del)
den <- ddmst(y,n,p,mean,cov,nu,del)
```

Description

Density and random generation for Multivariate Normal distributions with mean vector `mean`, and covariance matrix `cov`.

Usage

```
ddmvn(dat,n, p, mean, cov)
rdmvn( n, p, mean, cov)
```

Arguments

<code>dat</code>	An n by p numeric matrix, the dataset
<code>n</code>	An integer, the number of observations
<code>p</code>	An integer, the dimension of data
<code>mean</code>	A length of p vector, the mean
<code>cov</code>	A p by p matrix, the covariance

Value

`ddmvn` gives the density values; `rdmvn` generates the random numbers

See Also

[rdemmix](#), [ddmvn](#), [ddmsn](#), [ddmst](#), [rdmvn](#), [rdmsn](#), [rdmst](#).

Examples

```
n <- 100
p <- 2

mean <- rep(0,p)
cov  <- diag(p)

set.seed(3214)

y <- rdmvn( n,p,mean,cov)

den <- ddmvn(y,n,p,mean,cov)
```

Description

Density and random generation for Multivariate t-distributions with mean vector `mean`, covariance matrix `cov`, and degrees of freedom `nu`.

Usage

```
ddmvt(dat, n, p, mean, cov, nu)
rdmvt(    n, p, mean, cov, nu)
```

Arguments

<code>dat</code>	An n by p numeric matrix, the dataset
<code>n</code>	An integer, the number of observations
<code>p</code>	An integer, the dimension of data
<code>mean</code>	A length of p vector, the mean
<code>cov</code>	A p by p matrix, the covariance
<code>nu</code>	A positive number, the degrees of freedom

Value

`ddmvt` gives the density values; `rdmvt` generates the random numbers

See Also

[rdemmix](#), [ddmvn](#), [ddmsn](#), [ddmst](#), [rdmvn](#), [rdmsn](#), [rdmst](#).

Examples

```
n <- 100
p <- 2

mean   <- rep(0,p)
cov    <- diag(p)
nu     <- 3

set.seed(3214)

x     <- rdmvt(    n,p,mean,cov,nu)

den <- ddmvt(x ,n,p,mean,cov,nu)
```

<code>do.features</code>	<i>Extract the feature matrix</i>
--------------------------	-----------------------------------

Description

It is a pipeline function to get the features for each individual samples.

Usage

```
do.features(retpath=., p=4, g=2, distr="mst",
clusters=c("base", "mould"), data="0min",
panel="panel1", group="LNP_hi", samples,
markers=c("p-SFK", "BCL2", "CD20", "p-ERK"), class="jcm")
```

Arguments

<code>retpath</code>	A string of path to where the return files are stored.
<code>p</code>	The number of variables of data
<code>g</code>	The number of components of the mixture model
<code>distr</code>	A three letter string indicating the type of distribution to be fit. See Details.
<code>clusters</code>	A vector of strings for names of the clusters.
<code>data</code>	A string of the data name, which refers to a location where the panel data are stored.
<code>panel</code>	A string of the panel name, which refers to a subfolder where the real data are stored.
<code>group</code>	A string of clinical group.
<code>samples</code>	A vector of sample names.
<code>markers</code>	A vector of marker names.
<code>class</code>	A string of class, either "jcm" or space.

Value

"batch_features_data_panel_group.txt" "batch_features_data_panel_group.gct"

See Also

[do.jcm](#), [do.mix](#), [do.unzip](#)

Examples

```
## Not run:

info <- as.matrix(read.csv("JCM_clinical.csv"))

samples <- c(info[info[,2]=="sensitive",1])

data    <- "0min"
```

```

panel  <- "panel1"
group  <- "LNP_lo"
markers<- c("p-SFK","BCL2" , "CD20", "p-ERK")
do.features(retpath=.,p=4,g=2,distr="mst",
clusters=c("base","mould"),
data,panel,group,samples,markers,class="jcm")

## End(Not run)

```

do.jcm*Fit JCM on Each Sample of Panel***Description**

It is a pipeline function to fit each sample of a panel into JCM model.

Usage

```
do.jcm(data="4min",panel="panel1",samples,p=4,g=2,distr="mvt",itmax=100,
header=TRUE,fstype="CSV")
```

Arguments

data	A string of the data name, which refers to a location where the panel data are stored.
panel	A string of the panel name, which refers to a subfolder where the real data are stored.
samples	The vector of sample names
p	The number of variables of data
g	The number of components of the mixture model
distr	A three letter string indicating the type of distribution to be fit. See Details.
itmax	The maximum of iterations
header	A logical value, indicating whether there is a header in the data file.
fstype	A string indicating which type of format the data are stored, currently only "txt" and "csv" are handled.

Details

This function has to be run after domix,because the domix results are used as initial values.

Value

A ret file is returned for each sample. All estimated information of a sample are stored in its own ret file and can be accessed via dget() function in R.

See Also

[do.mix](#), [do.template](#), [do.unzip](#)

Examples

```
## Not run:

samples <- c(info[info[,2]=="LNP_lo",1])

do.jcm(data="4min", panel="panel1", samples, p=4, g=2,
distr="mvt", header=TRUE, ftype="CSV")

## End(Not run)
```

do.kld

Approximate Kullback-Leibler Distance

Description

Calculate the Approximate Kullback-Leibler Distance (KLD).

Usage

```
do.kld(datfile, SelfTemplate, ClassTemplates, st,
distr = "mvt", class = "jcm", header=TRUE, ftype="csv")
```

Arguments

datfile	A string where the data are stored.
SelfTemplate	A string where the individual template parameters are stored.
ClassTemplates	A string where the calss template parameters are stored.
st	A vector for the indices of the variables.
distr	A three letter string indicating component distribution: "mvn"=normal distribution, "mvt"=t-distribution,"msn"=skew normal distribution, "mst"=skew t-distribution.
class	A string, class="jcm", or not.
header	Are there headers in the data file.
ftype	The type of data file; default is "csv", alternative is "txt"

Value

A vector of KLD values

References

M.N., Do and M., Vetterli(2002). Wavelet-Based Texture Retrieval Using Generalized Gaussian Density and Kullback-Leibler Distance. IEEE Transactions on Image Processing, 11(2), 146-158.
T.M., Cover and J.A., Thomas(1991). Elements of Information Theory. New York: Wiley.

See Also

[entropy](#)

Examples

```

data(Lympho)

id=1

datname <- (Lympho$names)[id]

S    <- (Lympho$data)[[id]]

obj <- (Lympho$mvt)[[id]]

#
datfile <- paste(datname,".txt",sep="")
write.table(S,datfile, row.names=FALSE)

retfile <- paste(datname, ".ret", sep="")
dput(obj,retfile)

tmpfile <- paste((Lympho$template$names)[1:2], ".ret", sep="")

for(j in 1:2) {
  obj <- (Lympho$template$template)[[j]]
  dput(obj,tmpfile[j])
}

st <- 1:4

do.kld(datfile,retfile,tmpfile,st, distr = "mvt", class = "", ftype="txt")

```

do.mix

Fit the Multivariate Mixture Models on Each Sample of Panel

Description

It is a pipeline function to fit each sample of a panel into specified mixture model.

Usage

```
do.mix(data="4min",panel="panel1",samples,p=4,g=2,distr="mvt",
init=NULL,itmax=100,header=TRUE,ftype="CSV")
```

Arguments

data	A string of the data name, which refers to a location where the panel data are stored.
------	--

panel	A string of the panel name, which refers to a subfolder where the real data are stored.
samples	A vector of sample names
p	The number of variables of data
g	The number of components of the mixture model
distr	A three letter string indicating the type of distribution to be fit. See Details.
itmax	The maximum of iterations
init	The initial values
header	A logical value, indicating whether there is a header in the data file.
ftype	A string indicating which type of format the data are stored, currently only "txt" and "csv" are handled.

Details

The distribution type, determined by the `distr` parameter, which may take any one of the following values: "mvn" for a multivariate normal, "mvt" for a multivariate t-distribution, "msn" for a multivariate skew normal distribution and "mst" for a multivariate skew t-distribution.

This function fit the specified mixtrue model to all samples in the panel of data.

Value

A ret file is returned for each sample. All estimated information of a sample are stored in its own ret file and can be accessed via `dget()` function in R.

See Also

[do.jcm](#), [do.template](#), [do.unzip](#)

Examples

```
## Not run:

samples <- c(info[info[, 2]=="LNP_lo", 1])

do.mix(data="4min", panel="panel1", samples, p=4, g=2,
distr="mvt", header=TRUE, ftype="CSV")

## End(Not run)
```

Description

It is a pipeline function to get the mould component for each individual samples.

Usage

```
do.mould(data,panel,sample,p=4,g=2,distr="mst",class="")
```

Arguments

<code>data</code>	A string of the data name, which refers to a location where the panel data are stored.
<code>panel</code>	A string of the panel name, which refers to a subfolder where the real data are stored.
<code>sample</code>	A string of sample name.
<code>p</code>	The number of variables of data
<code>g</code>	The number of components of the mixture model
<code>distr</code>	A three letter string indicating the type of distribution to be fit. See Details.
<code>class</code>	A string of class, either "jcm" or space.

Details

This example is to illustrate how to plot the xyplot for multiple samples.

Value

A matrix of mould data.

See Also

[do.jcm](#), [do.mix](#), [do.unzip](#)

Examples

```
## Not run:

info <- as.matrix(read.csv("JCM_clinical.csv"))

samples <- c(info[info[,2]=="sensitive",1])

samples

# [1] "lp-j105"  "lp-j106"  "lp-j108"  "lp-j111"  "lp-j112"  "lp-j116"
# [7] "lp-j117"  "lp-j120"  "lp-j122"  "lp-j123"  "lp-j124"  "lp-j126"
#[13] "lp-j127"  "lp-j131a" "lp-j133a" "lp-j134"  "lp-j140"  "lp-j141"
# these are LNP- samples

data<- "4min"

panel<- "panel14"

distr<-"mst"
```

```

class<-""

#Row 1. 105, 108, 111, 112, 116;

lab3<- c("lp-j105","lp-j108","lp-j111","lp-j112","lp-j116")

z1=do.mould(data,panel,"lp-j105",p=4,g=2,distr,class)
z2=do.mould(data,panel,"lp-j108",p=4,g=2,distr,class)
z3=do.mould(data,panel,"lp-j111",p=4,g=2,distr,class)
z4=do.mould(data,panel,"lp-j112",p=4,g=2,distr,class)
z5=do.mould(data,panel,"lp-j116",p=4,g=2,distr,class)

#Row 2. 124, 126, 127, 134, 140;

lab2<- c("lp-j124","lp-j126","lp-j127","lp-j134","lp-j140")

y1=do.mould(data,panel,"lp-j124",p=4,g=2,distr,class)
y2=do.mould(data,panel,"lp-j126",p=4,g=2,distr,class)
y3=do.mould(data,panel,"lp-j127",p=4,g=2,distr,class)
y4=do.mould(data,panel,"lp-j134",p=4,g=2,distr,class)
y5=do.mould(data,panel,"lp-j140",p=4,g=2,distr,class)

samples <- c(info[info[,2]=="insensitive",1])

samples

# [1] "lp-j101"  "lp-j102"  "lp-j103"  "lp-j109"  "lp-j110"
#"lp-j114" [7] "lp-j119"  "lp-j121"  "lp-j125"  "lp-j128a"
# these are LNP+ samples

#Row 3. 109, 119, 128a, 121, 103

x1=do.mould(data,panel,"lp-j103",p=4,g=2,distr,class)
x2=do.mould(data,panel,"lp-j109",p=4,g=2,distr,class)
x3=do.mould(data,panel,"lp-j119",p=4,g=2,distr,class)
x4=do.mould(data,panel,"lp-j121",p=4,g=2,distr,class)
x5=do.mould(data,panel,"lp-j128a",p=4,g=2,distr,class)

lab1=c("lp-j103","lp-j109","lp-j119","lp-j121","lp-j128a")

dat <- rbind(x1 ,x2 ,x3 ,x4 ,x5 ,
y1 ,y2 ,y3 ,y4 ,y5 ,z1 ,z2 ,z3 ,z4 ,z5 )

markers <- c("p.PLcg2", "BCL2", "CD20", "p.STAT5")

colnames(dat) <-markers

labels <- factor(rep(1:15,
c(nrow(x1 ),nrow(x2 ),nrow(x3 ),nrow(x4 ),nrow(x5 ),
nrow(y1 ),nrow(y2 ),nrow(y3 ),nrow(y4 ),nrow(y5 ),
nrow(z1 ),nrow(z2 ),nrow(z3 ),nrow(z4 ),nrow(z5 ))),
labels=c(lab1,lab2,lab3))

```

```
#-----
jpeg("xyplotLPN_final(2).jpeg",height=2048*0.35,width=2048*.5)

xyplot(p.PLCg2~CD20|labels,layout=c(5,3),data=dat,
panel=function(x,y,...) {
xrange=range(dat[,1])+c(-0.65,0.5)
yrange=range(dat[,2])+c(-0.65,0.5)
Lab.palette <- colorRampPalette(c("blue", "orange", "red"),
space = "Lab")
panel.smoothScatter(x,y, colramp = Lab.palette,
range.x=list(xrange,yrange),bandwidth=c(0.1,0.1),...))

dev.off()

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

do.template *Find the distribution template*

Description

It is a pipeline function to get the distribution template for a particular group of samples.

Usage

```
do.template(data="4min",panel="panel1",group="LNP_lo",
distr="mvt",p=4,g=6,info,itmax=100,header=TRUE,ftype="csv")
JCM.template(data="4min",panel="panel1",group="LNP_lo",
distr="mvt",p=4,g=6,info,itmax=100,header=TRUE,ftype="csv")
```

Arguments

data	A string of the data name, which refers to a location where the panel data are stored.
panel	A string of the panel name, which refers to a subfolder where the real data are stored.
group	A string of clinical group.
p	The number of variables of data
g	The number of components of the mixture model
distr	A three letter string indicating the type of distribution to be fit. See Details.
info	The matrix of a two-column data, with variables as patient and condition group.
itmax	The maximum of iterations
header	A logical value, indicating whether there is a header in the data file.
ftype	A string indicating which type of format the data are stored, currently only "txt" and "csv" are handled.

Details

All samples are used to get the distribution template, while parts of samples are used to get initial values. JCM.template has to be run after function do.template since JCM.tmeplate input the results of do.template as initial values. All tempalte information are stored in the returned ret files which can be accessed by dget() in R.

Value

Two ret files are returned for each clinical group. One leads with "init", one with "template".

See Also

[do.jcm](#), [do.mix](#), [do.unzip](#)

Examples

```
## Not run:

info <- read.csv("JCM_clinical.csv")

do.template(data="4min", panel="panel1", group="LNP_lo",
distr="mvt", p=4, g=6, info=info, header=TRUE, ftype="csv")

JCM.template(data="4min", panel="panel1", group="LNP_lo",
distr="mvt", p=4, g=6, info=info, header=TRUE, ftype="csv")

## End(Not run)
```

do.unzip

Unzip the panel data

Description

It is a pipeline function to unzip the panel data into the project folder.

Usage

```
do.unzip(data, path=., exdir=.)
```

Arguments

data	A string referring to the zipped data name.
path	A string referring to a location where the zip files are stored.
exdir	A string referring to a where the unzipped data go..

Examples

```
## Not run:

data <- "TCR-6classes"
do.unzip(data, path=., exdir=.)

## End(Not run)
```

Description

As a main function, emmix fits the data into the specified multivariate mixture models via the EM Algorithm. Distributions (univariate and multivariate) available include Normal distribution, t-distribution, Skew Normal distribution, and Skew t-distribution.

Usage

```
emmix(dat, g, distr="mvn", ncov=3, clust=NULL, init=NULL, itmax=1000,
epsilon=1e-6, nkmeans=0, nrandom=10, nhclust=FALSE, debug=TRUE,
initloop=20)
```

Arguments

<code>dat</code>	The dataset, an n by p numeric matrix, where n is number of observations and p the dimension of data.
<code>g</code>	The number of components of the mixture model
<code>distr</code>	A three letter string indicating the type of distribution to be fitted, the default value is "mvn", the Normal distribution. See Details.
<code>ncov</code>	A small integer indicating the type of covariance structure; the default value is 3. See Details.
<code>clust</code>	A vector of integers specifying the initial partitions of the data; the default is NULL.
<code>init</code>	A list containing the initial parameters for the mixture model. See details. The default value is NULL.
<code>itmax</code>	A big integer specifying the maximum number of iterations to apply; the default value is 1000.
<code>epsilon</code>	A small number used to stop the EM algorithm loop when the relative difference between log-likelihood at each iteration become sufficient small; the default value is 1e-6.
<code>nkmeans</code>	An integer to specify the number of KMEANS partitions to be used to find the best initial values; the default value is 0.
<code>nrandom</code>	An integer to specify the number of random partitions to be used to find the best initial values; the default value is 10.

<code>nhclust</code>	A logical value to specify whether or not to use hierarchical cluster methods; the default is FALSE. If TRUE, the Complete Linkage method will be used.
<code>debug</code>	A logical value, if it is TRUE, the output will be printed out; FALSE silent; the default value is TRUE.
<code>initloop</code>	A integer specifying the number of initial loops when searching the best intial partitions.

Details

The distribution type, determined by the `distr` parameter, which may take any one of the following values: "mvn" for a multivariate normal, "mvt" for a multivariate t-distribution, "msn" for a multivariate skew normal distribution and "mst" for a multivariate skew t-distribution.

The covariance matrix type, represented by the `ncov` parameter, may be any one of the following: `ncov=1` for a common variance, `ncov=2` for a common diagonal variance, `ncov=3` for a general variance, `ncov=4` for a diagonal variance, `ncov=5` for $\sigma(h) * I(p)$ (diagonal covariance with same identical diagonal element values).

The parameter `init` requires following elements: `pro`, a numeric vector of the mixing proportion of each component; `mu`, a p by g matrix with each column as its corresponding mean; `sigma`, a three dimensional p by p by g array with its j th component matrix (p,p,j) as the covariance matrix for j th component of mixture models; `dof`, a vector of degrees of freedom for each component; `delta`, a p by g matrix with its columns corresponding to skew parameter vectors.

Since we treat the list of `pro,mu,sigma,dof`,and `delta` as a common structure of parameters for our mixture models, we need to include all of them in the initial parameter list `init` by default although in some cases it does not make sense, for example, `dof` and `delta` is not applicable to normal mixture model. But in most cases, the user only need give relevant paramters in the list.

When the parameter list `init` is given, the program ignores both initial partition `clust` and automatic partition methods such as `nkmeans`; only when both `init` and `clust` are not available, the program uses automatic approaches such as k-Means partition method to find the best initial values. All three automatic approaches are used to find the best initial partition and initial values if required.

The return values include all potential parameters `pro,mu,sigma,dof`,and `delta`, but user should not use or interpret irrelevant information arbitrarily. For example, `dof` and `delta` for Normal mixture models.

Value

<code>error</code>	Error code, 0 = normal exit; 1 = did not converge within <code>itmax</code> iterations; 2 = failed to get the initial values; 3 = singularity
<code>aic</code>	Akaike Information Criterion (AIC)
<code>bic</code>	Bayes Information Criterion (BIC)
<code>ICL</code>	Integrated Completed Likelihood Criterion (ICL)
<code>pro</code>	A vector of mixing proportions.
<code>mu</code>	A numeric matrix with each column corresponding to the mean.
<code>sigma</code>	An array of dimension (p,p,g) with first two dimension corresponding covariance matrix of each component.
<code>dof</code>	A vector of degrees of freedom for each component, see Details.
<code>delta</code>	A p by g matrix with each column corresponding to a skew parameter vector.
<code>clust</code>	A vector of final partition
<code>loglik</code>	The log likelihood at convergence

lk	A vector of log likelihood at each EM iteration
tau	An n by g matrix of posterior probability for each data point

References

- Biernacki C. Celeux G., and Govaert G. (2000). Assessing a Mixture Model for Clustering with the integrated Completed Likelihood. IEEE Transactions on Pattern Analysis and Machine Intelligence. 22(7). 719-725.
- McLachlan G.J. and Krishnan T. (2008). The EM Algorithm and Extensions (2nd). New Jersay: Wiley.
- McLachlan G.J. and Peel D. (2000). Finite Mixture Models. New York: Wiley.

See Also

[initEmmix](#), [rdemmmix](#).

Examples

```
#define the dimension of dataset

n1=300;n2=300;n3=400;
nn<-c(n1,n2,n3)

p <- 2
ng <- 3

#define the parameters
sigma<-array(0,c(2,2,3))
for(h in 2:3) sigma[,,h]<-diag(2)
sigma[,,1]<-cbind( c(1,0.2),c(0.2,1))
mu      <- cbind(c(4,-4),c(3.5,4),c( 0, 0))

#and other parameters if required for "mvt","msn","mst"
delta <- cbind(c(3,3),c(1,5),c(-3,1))
dof   <- c(3,5,5)

pro   <- c(0.3,0.3,0.4)

distr="mvn"
ncov=3

# generate a data set

set.seed(111) #random seed is reset

dat <- rdemmmix(nn,p,ng,distr,mu,sigma)

# the following code can be used to get singular data (remarked off)
# dat[1:300,2]<-4
# dat[300+1:300,1]<-2
## dat[601:1000,1]<-0
## dat[601:1000,2]<-0
```

```

#fit the data using KMEANS to get the initial partitions (10 trials)
obj <- emmix(dat,ng,distr,ncov,itmax=1000,epsilon=1e-5,nkmeans=10)

# alternatively, if we define initial values like
initobj<-list()

initobj$pro  <- pro
initobj$mu   <- mu
initobj$sigma<- sigma

initobj$dof  <- dof
initobj$delta<- delta

# then we can fit the data from initial values
obj <- emmix(dat,ng,distr,ncov,init=initobj,itmax=1000,epsilon=1e-5)

# finally, if we know initial partition such as
clust      <- rep(1:ng,nn)
# then we can fit the data from given initial partition
obj <- emmix(dat,ng,distr,ncov,clust=clust,itmax=1000,epsilon=1e-5)

# plot the 2D contour

# specify any two variable pair to plot
st <- c(1,2)

if(is.null(varnames <- dimnames(dat)[[2]]))
varnames<-paste("x",1:p,sep=)

## Not run:
emmix.contour.2d(dat[,st], obj$pro, obj$mu[st,], obj$sigma[st,st,],
obj$dof, obj$delta[st,], obj$clust, distr, grid=0.1,
levels=c(2,seq(5,80,by=5)),xlab=varnames[st[1]],ylab=varnames[st[2]])

## End(Not run)

```

Description

Contour of fitted mixture density can be an important indicator of goodness-of-fit. Together with the heatmaps, this function provides a general idea of how well the mixture model fits to the data.

Usage

```
emmix.flow(S, obj = NULL, distr="",diag.panel=TRUE,upper.panel="type2",
lower.panel ="type3", levels=seq(5, 95,by=20),attop=FALSE,clust=NULL,
title="",path="",plot=TRUE)
emmix.contours(S, obj = NULL, clust = NULL,distr="",diag.panel=TRUE,
upper.panel="type2",lower.panel="type3",levels=seq(5,95,by=20),
plot=TRUE,title="",path=,attop =FALSE)

emmix.filter(S, g=1,distr="mst",diag.panel=TRUE,upper.panel="type2",
lower.panel = "type3", levels = 90, attop = FALSE,
title="",path="",plot=TRUE)
```

Arguments

<code>g</code>	The number of components to filter the data at FSC and SSC channels.
<code>S</code>	A data frame.
<code>obj</code>	A list including the parameters of a (skew) mixture model.
<code>distr</code>	A three letter string specifying the distribution.
<code>diag.panel</code>	A logical value, plot density.
<code>upper.panel</code>	A string for the panel to be used in upper triangle.
<code>lower.panel</code>	A string for the panel to be used in lower triangle.
<code>levels</code>	A vector of contour percentage levels for the plots. It should be in the range of 0 to 100.
<code>attop</code>	A logical value indicating the direction of diagonal panels.
<code>clust</code>	A vector with the cluster labels for each observation of the sample.
<code>title</code>	The png file name.
<code>path</code>	The path to the folder where plots are stored.
<code>plot</code>	A logical variable, whether plot it in the windows.

Details

R package geneplotter is required.

Note

The input list 'obj' must be assigned a component "distr".

See Also

[conplot](#)

Examples

```
## Not run:
data(Lympho)
id=1
```

```

S    <- (Lympho$data)[[id]]

obj <- (Lympho$mvt)[[id]]

datname <- (Lympho$names)[id]

x11()
emmix.flow(S,obj,diag.panel=TRUE)

x11()
emmix.contours(S,obj)

g=5
p=4
distr="mvn"

obj <- emmix(S,g,distr)

x11()
emmix.flow(S,obj,diag.panel=TRUE)

x11()
emmix.contours(S,obj)

## End(Not run)

```

emmixfit*Fit the Multivariate Skew Mixture Models***Description**

The engines to fit the data into mixture models using initial partition or initial values. set.

Usage

```

emmixfit1(dat, g, clust,      distr, ncov, itmax, epsilon, initloop=20)
emmixfit2(dat, g,           init, distr, ncov, itmax, epsilon)

```

Arguments

dat	The dataset, an n by p numeric matrix, where n is number of observations and p the dimension of data.
g	The number of components of the mixture model
distr	A three letter string indicating the type of distribution to be fit. See Details.
ncov	A small integer indicating the type of covariance structure. See Details.
clust	A vector of integers specifying the initial partitions of the data
init	A list containing the initial parameters for the mixture model. See details.
itmax	A big integer specifying the maximum number of iterations to apply

epsilon	A small number used to stop the EM algorithm loop when the relative difference between log-likelihood at each iteration become sufficient small.
initloop	A integer specifying the number of initial loops

Details

The distribution type, determined by the `distr` parameter, which may take any one of the following values: "mvn" for a multivariate normal, "mvt" for a multivariate t-distribution, "msn" for a multivariate skew normal distribution and "mst" for a multivariate skew t-distribution.

The covariance matrix type, represented by the `ncov` parameter, may be any one of the following: `ncov=1` for a common variance, `ncov=2` for a common diagonal variance, `ncov=3` for a general variance, `ncov=4` for a diagonal variance, `ncov=5` for $\text{sigma}(h) * I(p)$ (diagonal covariance with same identical diagonal element values).

The parameter `init` is a list with elements: `pro`, a numeric vector of the mixing proportion of each component; `mu`, a p by g matrix with each column as its corresponding mean; `sigma`, a three dimensional p by p by g array with its j th component matrix (p,p,j) as the covariance matrix for j th component of mixture models; `dof`, a vector of degrees of freedom for each component; `delta`, a p by g matrix with its columns corresponding to skew parameter vectors.

Value

error	Error code, 0 = normal exit; 1 = did not converge within <code>itmax</code> iterations; 2 = failed to get the initial values; 3 = singularity
aic	Akaike Information Criterion (AIC)
bic	Bayes Information Criterion (BIC)
pro	A vector of mixing proportions, see Details.
mu	A numeric matrix with each column corresponding to the mean, see Details.
sigma	An array of dimension (p,p,g) with first two dimension corresponding covariance matrix of each component, see Details.
dof	A vector of degrees of freedom for each component, see Details.
delta	A p by g matrix with each column corresponding to a skew parameter vector, see Details.
clust	A vector of final partition
loglik	The loglikelihood at convergence
lk	A vector of loglikelihood at each EM iteration
tau	An n by g matrix of posterior probability for each data point

References

- McLachlan G.J. and Krishnan T. (2008). The EM Algorithm and Extensions (2nd). New Jersay: Wiley.
 McLachlan G.J. and Peel D. (2000). Finite Mixture Models. New York: Wiley.

See Also

`init.mix, initEmmix, emmix, rdemmix, rdemmix2, rdmvn, rdmvt, rdmsn, rdmst.`

Examples

```

n1=300;n2=300;n3=400;
nn <-c(n1,n2,n3)
n=1000
p=2
ng=3

sigma<-array(0,c(2,2,3))
for(h in 2:3) sigma[,,h]<-diag(2)
sigma[,,1]<-cbind( c(1,0),c(0,1))
mu <- cbind(c(4,-4),c(3.5,4),c( 0, 0))

# for other distributions,
#delta <- cbind(c(3,3),c(1,5),c(-3,1))
#dof    <- c(3,5,5)

pro   <- c(0.3,0.3,0.4)

distr="mvn"
ncov=3

#first we generate a data set
set.seed(111) #random seed is set
dat <- rdemmix(nn,p,ng,distr,mu,sigma,dof=NULL,delta=NULL)

#start from initial partition
clust<- rep(1:ng,nn)
obj1 <- emmixfit1(dat, ng, clust, distr, ncov, itmax=1000, epsilon=1e-4)

#start from initial values
#alternatively, if we define initial values like

init<-list()

init$pro<-pro
init$mu<-mu
init$sigma<-sigma

# for other distributions,
#delta <- cbind(c(3,3),c(1,5),c(-3,1))
#dof    <- c(3,5,5)
#init$dof<-dof
#init$delta<-delta

obj2 <- emmixfit2(dat, ng, init, distr, ncov,itmax=1000, epsilon=1e-4)

```

Description

Calculate the mode points for each component of skew mixture models.

Usage

```
emmixMOD(p,g,distr,mu,sigma,dof,delta,nrand=10000)
```

Arguments

p	The dimension of the data
g	The number of components to be fit
distr	A three letter string of distribution id
mu	A numeric matrix with each column corresponding to the mean
sigma	An array of dimension (p,p,g) with first two dimensions corresponding covariance matrix of each component
dof	A vector of degrees of freedom for each component
delta	A matrix with each column as skew parameter vector
nrand	The number of random numbers.

Value

A p by g matrix of mode points for each component of the skew mixture model.

Examples

```
p=2
g=3

sigma<-array(0,c(2,2,3))
for(h in 2:3) sigma[,,h]<-diag(2)
sigma[,,1]<-cbind( c(1,0),c(0,1))
mu <- cbind(c(4,-4),c(3.5,4),c( 0, 0))

delta <- cbind(c(3,3),c(1,5),c(-3,1))
dof    <- c(3,5,5)

distr="mst"

emmixMOD(p,g,distr,mu,sigma,dof,delta,nrand=10000)
```

Description

Calculate the entropy of data for a given mixture template.

Usage

```
entropy(dat, distr, class, obj)
```

Arguments

<code>dat</code>	A data frame.
<code>distr</code>	A three letter string indicating component distribution, "mvn"=normal distribution, "mvt"=t-distribution,"msn"=skew normal distribution, "mst"=skew t-distribution.
<code>class</code>	A string, class="jcm", or not.
<code>obj</code>	A list of template parameters.

Value

A positive value.

References

- M.N., Do and M., Vetterli(2002). Wavelet-Based Texture Retrieval Using Generalized Gaussian Density and Kullback-Leibler Distance. IEEE Transactions on Image Processing, 11(2), 146-158.
 T.M., Cover and J.A., Thomas(1991). Elements of Information Theory. New York: Wiley.

See Also

[do.kld](#)

`error.rate`

Error Rate

Description

Calculate the Error Rate of a partition

Usage

```
error.rate(clust1,clust2)
F.Measures(Clusters,Klabels)
```

Arguments

<code>clust1</code>	An integer vector of cluster label 1
<code>clust2</code>	An integer vector of cluster label 2
<code>Clusters</code>	An integer vector of the true membership labels
<code>Klabels</code>	An integer vector of the predicted labels

Details

`clust1` and `clust 2` must match, i.e, same number of clusters

Value

error.rate gives Error Rate

Examples

```
clu1<-c(1,2,3,1,1,2,2,3,3)
clu2<-c(2,2,2,1,1,1,3,3,3)
error.rate(clu1, clu2)
```

getcov

*Covariance***Description**

Recalculate covariance for a given structure.

Usage

```
getcov(msigma, sumtau, n, p, g, ncov)
```

Arguments

<code>msigma</code>	An array for the covariance matrices.
<code>sumtau</code>	A vector for the expected number of observations from each component.
<code>n</code>	An integer for the sample size.
<code>p</code>	An integer for the dimension.
<code>g</code>	An integer for the number of components.
<code>ncov</code>	An integer for type of covariance.

Value

The covariance array.

References

McLachlan G.J. and Krishnan T. (2008). The EM Algorithm and Extensions (2nd). Hoboken, New Jersey: Wiley.

McLachlan G.J. and Peel D. (2000). Finite Mixture Models. New York: Wiley.

getICL

*The ICL criterion***Description**

Calculate the Integrated Completed Likelihood(ICL) criterion

Usage

```
getICL(x, n, p, g, distr, ncov, pro, mu, sigma, dof, delta, clust)
```

Arguments

x	An n by p data matrix
n	The total number of points
p	Dimension of data
g	the number of components of the mixture model
distr	A three letter string indicating the type of distribution to be fit.
ncov	A small integer indicating the type of covariance structure.
pro	A vector of mixing proportions
mu	A numeric matrix with each column corresponding to the mean
sigma	An array of dimension (p,p,g) with first two dimension corresponding covariance matrix of each component
dof	A vector of degrees of freedom for each component
delta	A p by g matrix with each column corresponding to a skew parameter vector
clust	A vector of partition

Value

ICL	ICL value
-----	-----------

References

Biernacki C. Celeux G., and Govaert G. (2000). Assessing a Mixture Model for Clustering with the integrated Completed Likelihood. IEEE Transactions on Pattern Analysis and Machine Intelligence. 22(7). 719-725.

Examples

```
n1=300;n2=300;n3=400;
nn <-c(n1,n2,n3)
n=1000
p=2
ng=3

sigma<-array(0,c(2,2,3))
for(h in 2:3) sigma[,,h]<-diag(2)
sigma[,,1]<-cbind( c(1,0),c(0,1))
```

```

mu  <- cbind(c(4,-4),c(3.5,4),c( 0, 0))

pro   <- c(0.3,0.3,0.4)

distr="mvn"
ncov=3

#first we generate a data set
set.seed(111) #random seed is set
dat <- rdemmix(nn,p,ng,distr,mu,sigma,dof=NULL,delta=NULL)

#start from initial partition
clust<- rep(1:ng,nn)
obj <- emmixfit1(dat, ng, clust, distr, ncov, itmax=1000,epsilon=1e-4)

getICL(dat,n,p,ng, distr,ncov,obj$pro,obj$mu,obj$sigma,obj$dof,
obj$delta,obj$clust)

```

initEmmix*Initialize Emmix Parameters***Description**

Obtains intial parameter set for use in the EM algorithm. Grouping of the data occurs through one of three possible clustering methods: k-means, random start, and hierarchical clustering.

Usage

```
initEmmix(dat, g, clust, distr, ncov,maxloop=20)
init.mix( dat, g, distr, ncov, nkmeans, nrandom, nhclust,maxloop=20)
```

Arguments

dat	The dataset, an n by p numeric matrix, where n is number of observations and p the dimension of data.
g	The number of components of the mixture model
distr	A three letter string indicating the type of distribution to be fit. See Details.
ncov	A small integer indicating the type of covariance structure. See Details.
clust	An initial partition of the data
nkmeans	An integer to specify the number of KMEANS partitions to be used to find the best initial values
nrandom	An integer to specify the number of random partitions to be used to find the best initial values
nhclust	A logical value to specify whether or not to use hierarchical cluster methods. If TRUE, the Complete Linkage method will be used.
maxloop	An integer to specify how many iterations to be tried to find the initial values,the default value is 10.

Details

The distribution type, determined by the `distr` parameter, which may take any one of the following values: "mvn" for a multivariate normal, "mvt" for a multivariate t-distribution, "msn" for a multivariate skew normal distribution and "mst" for a multivariate skew t-distribution.

The covariance matrix type, represented by the `ncov` parameter, may be any one of the following: `ncov=1` for a common variance, `ncov=2` for a common diagonal variance, `ncov=3` for a general variance, `ncov=4` for a diagonal variance, `ncov=5` for $\sigma(h) * I(p)$ (diagonal covariance with same identical diagonal element values).

The return values include following components: `pro`, a numeric vector of the mixing proportion of each component; `mu`, a p by g matrix with each column as its corresponding mean; `sigma`, a three dimensional p by p by g array with its j th component matrix (p,p,j) as the covariance matrix for j th component of mixture models; `dof`, a vector of degrees of freedom for each component; `delta`, a p by g matrix with its columns corresponding to skew parameter vectors.

When the dataset is huge, it becomes time-consuming to use a large maxloop to try every initial partition. The default is 10. During the procedure to find the best initial clustering and initial values, for t-distribution and skew t-distribution, we don't estimate the degrees of freedom `dof`, instead they are fixed at 4 for each component.

Value

<code>pro</code>	A vector of mixing proportions, see Details.
<code>mu</code>	A numeric matrix with each column corresponding to the mean, see Details.
<code>sigma</code>	An array of dimension (p,p,g) with first two dimension corresponding covariance matrix of each component, see Details.
<code>dof</code>	A vector of degrees of freedom for each component, see Details.
<code>delta</code>	A p by g matrix with each column corresponding to a skew parameter vector, see Details.

References

- McLachlan G.J. and Krishnan T. (2008). The EM Algorithm and Extensions (2nd). New Jersey: Wiley.
 McLachlan G.J. and Peel D. (2000). Finite Mixture Models. New York: Wiley.

See Also

[emmix](#)

Examples

```
sigma<-array(0,c(2,2,3))
for(h in 2:3) sigma[,,h]<-diag(2)
sigma[,,1]<-cbind(c(1,0.2),c(0.2,1))
mu <- cbind(c(4,-4),c(3.5,4),c( 0, 0))
delta <- cbind(c(3,3),c(1,5),c(-3,1))
dof   <- c(3,5,5)
pro   <- c(0.3,0.3,0.4)
n1=300;n2=300;n3=400;
nn<-c(n1,n2,n3)
n=1000
p=2
```

```

ng=3
distr="mvn"
ncov=3
#first we generate a data set
set.seed(111) #random seed is set
dat <- rdemmix(nn,p,ng,distr,mu,sigma,dof,delta)
clust<- rep(1:ng,nn)
initobj1 <- initEmmix(dat,ng,clust,distr, ncov)
initobj2 <- init.mix( dat,ng,distr,ncov,nkmeans=10,nrandom=0,nhclust=FALSE)

```

inverse*Inverse of a covariance matrix***Description**

Calculate the inverse of a covariance matrix.

Usage

```
inverse(sigma, p)
```

Arguments

sigma	The covariance matrix.
p	The dimension of the matrix.

Value

The inverse of the covariance matrix.

Note

The covariance matrix may be singular. This is of use only for the clustering of the data.

Author(s)

Kui Wang

Examples

```

a<- matrix(c(1,0,0,0),ncol=2)
a
inverse(a,2)

```

Description

Jointly clustering and matching multiple samples

Usage

```
jcamst(datfiles, p, g, initobj, ncov = 3, dofon = 0, debug = 1,
itmax = 1000, epsilon = 1e-05, header = TRUE, ftype="csv")
jcamvt(datfiles, p, g, initobj, ncov = 3, dofon = 0, debug = 1,
itmax = 1000, epsilon = 1e-05, header = TRUE, ftype="csv")
```

Arguments

datfiles	Full path to where the data are stored.
p	The dimension of data.
g	The number of components.
initobj	The list of initial parametrers.
ncov	The type of covariance structure.
dofon	When set, the degrees of freedom for each component will be same.
debug	Whether to print log-likelihood for each iteration.
itmax	The maximum number of iterations.
epsilon	Stopping rule based on relative change in the log-likelihood.
header	Are there headers in the data file.
ftype	A string indicating which type of format the data are stored, currently only "txt" and "csv" are handled.

Value

error	Error code, 0 = normal exit; 1 = did not converge within itmax iterations.
loglik	The log likelihood at convergence.
bic	Bayesian Information Criterion (BIC).
pro	A vector with the mixing proportions.
alfa	A numeric matrix with each column corresponding to the mean.
mu	A numeric matrix with each column corresponding to the mean.
sigma	An array of dimension (p,p,g) with first two arguments corresponding to the covariance matrix of each component.
dof	A vector of degrees of freedom for each component.
delta	A p by g matrix with each column corresponding to a skew parameter vector.
theta	A matrix with each column corresponding to the variances of the random effects item a.
thetu	A vector with the variance of the random effects item b.
distr	A three letter string standing for distribution.
class	Class should be "jcm".

See Also

[msmv](#) [msmvn](#) [emmix](#).[flow](#) [jcamst](#).[predict](#)

Examples

```
data(Lympho)

# we calculate the template of class 0min.

datnames <- (Lympho$names)[1:5]

# make a temp dir for data files

system("mkdir templym")

datfiles <- paste("templym/",datnames,".txt",sep=)

for(id in 1:5) {

  S   <- (Lympho$data)[[id]]
  write.table(S,datfiles[id],row.names=FALSE)

}

# set initial values

init<- Lympho$init

p <- ncol(S)
g <- length(init$pro)

init$theta <- array(0.1,c(p,g))
init$thetu <- rep(0.1,g)

#do individuals
## Not run:
obj <- jcamvt(datfiles[1],p,g,init,itmax=100,ftype="txt")
obj <- jcamst(datfiles[1],p,g,init,itmax=100,ftype="txt")

## End(Not run)

# do class template
## Not run:

obj <- jcamvt(datfiles,p,g,init,itmax=100,ftype="txt")
obj <- jcamst(datfiles,p,g,init,itmax=100,ftype="txt")

## End(Not run)

# Do JCM MST alternatively
## Not run:

id=1
```

```

S    <- (Lympho$data)[[id]]

p=4
g=4
n <- nrow(S)

s1 <- sample(1:n,min(2000,n))
x11()
emmix.flow(S[s1,],diag.panel=TRUE)

init <- emmix(S[s1,],g,"mst",itmax=100)

init$distr="mst"

emmix.flow(S[s1,],init,diag.panel=TRUE)

init$theta<- array(0.1,c(p,g))
init$thetu<- rep(0.1,g)

obj <- jcamst(datfiles[1],p,g,init,itmax=100,fstype="txt")

pre <- jcamst.predict(S, g, obj$pro, obj$mu, obj$sigma,
obj$dof, obj$delta,obj$theta, obj$thetu)

#plot the aligned data
x11()
emmix.flow(pre$eee,diag.panel=TRUE)
title(main = list(paste("Aligned flow sample:",datnames[id]),
cex=1.5,col="blue", font=3))

#plot the data
x11()
emmix.flow(S,clust=pre$clust,diag.panel=TRUE)
title(main = list(paste("flow sample:",datnames[id]),
cex=1.5,col="blue", font=3))

## End(Not run)

```

jcamst.predict*Calculate the aligned values (via JCAMST)***Description**

Using the estimated parameters to calculate the aligned values at each data points, as well as the posterior probability and final partitions of the data.

Usage

```
jcamst.predict(dat,g,pro,mu,sigma,dof,delta,theta,thetu)
jcamvt.predict(dat,g,pro,mu,sigma,dof,theta,thetu)
```

Arguments

<code>dat</code>	An n by p numeric matrix, the dataset
<code>g</code>	The number of clusters
<code>pro</code>	A vector of mixing proportions, see Details.
<code>mu</code>	A numeric matrix with each column corresponding to the mean, see Details.
<code>sigma</code>	An array of dimension (p,p,g) with first two dimension corresponding covariance matrix of each component, see Details.
<code>dof</code>	A vector of degrees of freedom for each component, see Details.
<code>delta</code>	A p by g matrix with each column corresponding to a skew parameter vector, see Details.
<code>theta</code>	A p by g matrix with each column corresponding to the variance of the scaling random effect.
<code>thetu</code>	A vector of variance for each component.

Value

`eee` is a n by p matrix giving the aligned values; `tau` is a n by g matrix of the posterior probability; `clust` is a vector of the clustering.

See Also

[jcamvt](#). [jcamst](#).

Examples

```
## Not run:

# get the data

dat <- read.table("mydata.txt",header=T)

# get the estimated parameters of JCAMVT model
obj<- dget("xxx.ret")

# calculate the predicted values

pred <- predict.jcamst(dat,length(obj$pro),obj$pro,obj$mu,
obj$sigma,obj$dof,obj$delta,obj$theta,obj$thetu)

# plot the clustering results in 2D pairs
# using the aligned values
plot((pred$eee)[,c(1,2)],col=pred$clust)

## End(Not run)
```

Lympho

Lymphocytes

Description

Lympho consists of ten samples of flow cytometric data, as well as fitting information for msrv and jcavt.

Usage

```
data(Lympho)
```

Format

A list of 6 components:

\$names: names of the ten samples;

\$data: list of ten data frames;

\$mvt: list of MVT templates for the ten individual samples;

\$init: initial values;

\$jcm: list of JCM templates for the ten individual samples;

\$template: a list with two components, \$names and \$template; \$names gives four class template names,"0min-mvt", "5min-mvt", "0min-jcm" and "5min-jcm"; \$template is the list of four class templates.

Source

http://www.broadinstitute.org/cancer/software/genepattern/modules/FLAME/published_data.html

Examples

```
data(Lympho)
```

msrv

Distribution template

Description

Calculate the distribution template of a class of multiple samples

Usage

```
msmvt(datfiles, p, g, initobj, ncov = 3, common = 0, debug = 1,
itmax = 1000, epsilon = 1e-05, header = TRUE,ftype="txt")

msmvn(datfiles, p, g, initobj, ncov = 3, debug = 1,
itmax = 1000, epsilon = 1e-05, header = TRUE,ftype="txt")

msmsn(datfiles, p, g, initobj, ncov = 3, debug = 1,
itmax = 1000, epsilon = 1e-05, header = TRUE,ftype="txt")

msmst(datfiles, p, g, initobj, ncov = 3, debug = 1,
itmax = 1000, epsilon = 1e-05, header = TRUE,ftype="txt")
```

Arguments

datfiles	Full path to where the data are stored.
p	Dimension of data.
g	The number of components.
initobj	The list of initial parametrers.
ncov	The type of covariance structure.
common	When set, the degrees of freedom for all components will be same.
debug	When set, the log likelihood for each iteration will be printed.
itmax	The maximum number of iterations.
epsilon	Stopping rule based on the relative change in the log likelihood.
header	Are there headers in the data files.
ftype	The type of data file; default is "txt", alternative is "csv"

Value

error	Error code, 0 = normal exit; 1 = did not converge within itmax iterations.
loglik	The log likelihood at convergence.
bic	Bayesian Information Criterion (BIC).
pro	A vector with the mixing proportions.
mu	A numeric matrix with each column corresponding to the mean.
sigma	An array of dimension (p,p,g) with first two arguments corresponding to the covariance matrix of each component.
dof	A vector of degrees of freedom for each component, see Details.
delta	A p by g matrix of zeros.
distr	A three letter string indicating the distribution.

References

McLachlan G.J. and Krishnan T. (2008). The EM Algorithm and Extensions (2nd). Hoboken, New Jersey: Wiley.

McLachlan G.J. and Peel D. (2000). Finite Mixture Models. New York: Wiley.

See Also

[emmix.flow](#)

Examples

```
data(Lympho)

# we calculate the template of class 0min.

datnames <- (Lympho$names)[1:5]

# make a temp dir for data files

system("mkdir templym")

datfiles <- paste("templym/",datnames,".txt",sep=)

for(id in 1:5) {

  S   <- (Lympho$data)[[id]]
  write.table(S,datfiles[id],row.names=FALSE)

}

# set initial values

init <- Lympho$init

# do class template

## Not run:

obj1 <- msmvt(datfiles,p,g,init,itmax=100)

obj2 <- msmvn(datfiles,p,g,init,itmax=100)

## End(Not run)
```

Description

Calculate the degrees of freedom

Usage

```
mvt.dof(sumtau, sumlnv, lx = 2+1e-04, ux = 200)
```

Arguments

sumtau	A quantity.
sumlnv	A quantity.
lx	Lower limit.
ux	Upper limit.

Details

It is called by msmvt, jcamvt and jcamst. When there is no solution between lx and ux, ux will be returned.

Value

degrees of freedom.

Note

It is called by msmvt, jcamst and jcamvt.

plotmixt.skew

*Overlaid contours***Description**

It is a function to plot overlaid multiple distribution templates.

Usage

```
plotmixt.skew(retfiles,which.dim,distr="mvt",
  cont=c(25,50,75), xlim, ylim, zlim, gridsize,
  nrand=1e5, var.label, line.col, marginal.col,
  feature.col, lwd=1,...)
```

Arguments

retfiles	The return of dir()
which.dim	which dimensions are to be plotted.
distr	A three letter string indicating the type of distribution to be fit. See Details.
cont	The levels of contours in percentage.
xlim	The range of the first variable
ylim	The range of the second variable
zlim	The range of the third variable
gridsize	The grid size
nrand	The number of random number to be generated.
var.label	The labels of variables
line.col	The colors of contour lines
marginal.col	The color of lines
feature.col	The color of features
lwd	The width of lines
...	others

Details

The source code is stored in R program "JCM_OverlayPlots.R".

See Also

[emmix](#)

Examples

```
## Not run:

setwd("../Desktop/JCM_final")
require(emmix)

splom.settings <- list(layout.heights=list(top.padding=0.5,
main.key.padding=0, key.axis.padding=0, axis.xlab.padding=0,
xlab.key.padding=0, key.sub.padding=0, bottom.padding=0, strip=0),
layout.widths=list(left.padding=0, key.ylab.padding=0,ylab.axis.padding=0,
axis.key.padding=0,right.padding=0.5))

labels = c("SLP76" , "ZAP70", "CD4", "CD45RA")

whichcols = c(3,2,4)

retfiles<- paste( c(0,1,3), "min-jca-mvt-5.ret", sep="")

col5=rainbow(5)[c(1,3,5)]

set.seed(8192)

#1. code for "3_timepoints.jpeg"

jpeg(filename="3_timepoints.jpeg", height=512, width=512)

plotmixt.skew(retfiles, which.dim=whichcols, distr="mvt",
xlim=c(2,10), ylim=c(2,10), zlim=c(2,10),
par.settings=splom.settings, nrand=1e4,
var.label=labels[whichcols], lwd=1, cont=c(50,90),
line.col=col5, add.feature=TRUE, feature.col=col5,
key=list(space=list("bottom")),
text=list(c("0 min", "1 min", "3 min")),
lines=list(lwd=1, col=col5), columns=3))

dev.off()

#2. 3sample plots

dirpath <- dir("d:/uq/Data/SixPanel",full.names=T)

pid <- 3 #1 min class

files <- dir(dirpath[pid],full.names=T)

dat1 <- read.table(files[1],header=T)
```

```

dat2 <- read.table(files[2],header=T)
dat3 <- read.table(files[3],header=T)

dat <- rbind(dat1,dat2,dat3)

p=4
g=5

retAll <- emmix(dat,g,"mvt")

init1 <- emmix(dat1,g,"mvt",init=retAll)
init2 <- emmix(dat2,g,"mvt",init=retAll)
init3 <- emmix(dat3,g,"mvt",init=retAll)

filenames <- dir(dirpath[pid])

dput(init1,paste(filenames[1], ".ret", sep=""))
dput(init2,paste(filenames[2], ".ret", sep=""))
dput(init3,paste(filenames[3], ".ret", sep=""))

dput(retAll,"1min-mvt-5.ret")

#-----

dirpath <- dir("d:/uq/Data/SixPanel",full.names=T)

pid <- 3 #1 min class

filenames <- dir(dirpath[pid])

labels = c("SLP76" , "ZAP70", "CD4", "CD45RA")

whichcols = c(3,2,4)

retfiles<-c(paste(filenames,".ret",sep=""), paste( "1min-mvt-5.ret", sep=""))

retfiles<-c(paste( "1min-mvt-5a.ret", sep=""), paste( "1min-mvt-5.ret", sep=""))

pa <- c(rainbow(5)[c(1,3,5)],"blue")

set.seed(12345)

jpeg("1min(three samples).jpeg",width=512,height=512)

plotmixt.skew(retfiles, whichcols, distr="mvt",
xlim=c(1.5,8), ylim=c(1.5,8), zlim=c(1.5,10),
par.settings=splom.settings, nrand=10000,
var.label=labels[whichcols], lwd=1,
line.col=pa, cont=c(50,90), key=list(space=list("bottom")),
text=list(c("sample1", "sample2", "sample3","template")),
lines=list(lwd=1, col=pa), columns=4))

dev.off()

## End(Not run)

```

rdemmix*Simulate Data Using Mixture Models*

Description

Generate random number from specified mixture models, including univariate and multivariate Normal distribution, t-distribution, Skew Normal distribution, and Skew t-distribution.

Usage

```
rdemmix(nvect,p,g,distr,    mu,sigma,dof=NULL,delta=NULL)
rdemmix2(n,    p,g,distr,pro,mu,sigma,dof=NULL,delta=NULL)
rdemmix3(n,    p,g,distr,pro,mu,sigma,dof=NULL,delta=NULL)
```

Arguments

nvect	A vector of how many points in each cluster,c(n1,n2,..,ng)
n	The total number of points
p	Dimension of data
g	The number of clusters
distr	A three letter string indicating the distribution type
pro	A vector of mixing proportions, see Details.
mu	A numeric matrix with each column corresponding to the mean, see Details.
sigma	An array of dimension (p,p,g) with first two dimension corresponding covariance matrix of each component, see Details.
dof	A vector of degrees of freedom for each component, see Details.
delta	A p by g matrix with each column corresponding to a skew parameter vector, see Details.

Details

The distribution type, determined by the `distr` parameter, which may take any one of the following values: "mvn" for a multivariate normal, "mvt" for a multivariate t-distribution, "msn" for a multivariate skew normal distribution and "mst" for a multivariate skew t-distribution. `pro`, a numeric vector of the mixing proportion of each component; `mu`, a p by g matrix with each column as its corresponding mean; `sigma`, a three dimensional p by p by g array with its j th component matrix (p,p,j) as the covariance matrix for j th component of mixture models; `dof`, a vector of degrees of freedom for each component; `delta`, a p by g matrix with its columns corresponding to skew parameter vectors.

Value

both `rdemmix` and `rdemmix2` return an n by p numeric matrix of generated data;
`rdemmix3` gives a list with components `data`, the generated data, and `cluster`, the clustering of data.

References

- McLachlan G.J. and Krishnan T. (2008). The EM Algorithm and Extensions (2nd). New Jersay: Wiley.
- McLachlan G.J. and Peel D. (2000). Finite Mixture Models. New York: Wiley.

See Also

[rdmvn](#),[rdmvt](#),[rdmsn](#), [rdmst](#).

Examples

```
#specify the dimension of data, and number of clusters
#the number of observations in each cluster
n1=300;n2=300;n3=400;
nn<-c(n1,n2,n3)

p=2
g=3

#specify the distribution
distr <- "mvn"

#specify mean and covariance matrix for each component

sigma<-array(0,c(2,2,3))
for(h in 2:3) sigma[,,h]<-diag(2)
sigma[,,1]<-cbind( c(1,-0.1),c(-0.1,1))
mu <- cbind(c(4,-4),c(3.5,4),c( 0, 0))

#reset the random seed
set.seed(111)
#generate the dataset
dat <- rdemmix(nn,p,g,distr,      mu,sigma)

# alternatively one can use
pro <- c(0.3,0.3,0.4)
n=1000
set.seed(111)
dat <- rdemmix2(n,p,g,distr,pro,mu,sigma)
plot(dat)

# and

set.seed(111)
dobj <- rdemmix3(n,p,g,distr,pro,mu,sigma)
plot(dobj$data)

#other distributions such as "mvt","msn", and "mst".

#t-distributions
```

```
dof      <- c(3,5,5)
dat <- rdemmix2(n,p,g,"mvt",pro,mu,sigma,dof)
plot(dat)

#Skew Normal distribution
delta <- cbind(c(3,3),c(1,5),c(-3,1))
dat <- rdemmix2(n,p,g,"msn",pro,mu,sigma,delta=delta)
plot(dat)

#Skew t-distribution
dat <- rdemmix2(n,p,g,"mst",pro,mu,sigma,dof,delta)
plot(dat)
```

Index

*Topic **cluster**

- bootstrap, 2
- ddmix, 5
- ddmsn, 7
- ddmst, 8
- ddmvn, 9
- ddmvt, 10
- emmix, 20
- emmixfit, 25
- emmixMOD, 27
- error.rate, 29
- getICL, 31
- initEmmix, 32
- jcamst.predict, 37
- rdemmix, 45

*Topic **datasets**

- bootstrap, 2
- ddmsn, 7
- ddmst, 8
- ddmvn, 9
- ddmvt, 10
- emmix, 20
- emmixfit, 25
- emmixMOD, 27
- error.rate, 29
- getICL, 31
- initEmmix, 32
- Lympho, 39
- rdemmix, 45

bootstrap, 2

conplot, 4, 24

conplot2 (conplot), 4

conplot3 (conplot), 4

ddmix, 5

ddmsn, 6, 7, 8–10

ddmst, 6, 7, 8, 9, 10

ddmvn, 6–8, 9, 10

ddmvt, 6–9, 10

do.features, 11

do.jcm, 11, 12, 15, 16, 19

do.kld, 13, 29

do.mix, 11, 13, 14, 16, 19

do.mould, 15

do.template, 13, 15, 18

do.unzip, 11, 13, 15, 16, 19, 19

emmix, 3, 20, 26, 33, 43

emmix.contours, 23

emmix.filter (emmix.contours), 23

emmix.flow, 5, 36, 41

emmix.flow (emmix.contours), 23

emmixfit, 25

emmixfit1 (emmixfit), 25

emmixfit2 (emmixfit), 25

emmixMOD, 27

entropy, 14, 28

error.rate, 29

F.Measures (error.rate), 29

getcov, 30

getICL, 31

init.mix, 26

init.mix (initEmmix), 32

initEmmix, 22, 26, 32

inverse, 34

jcamst, 35, 38

jcamst.predict, 36, 37

jcamvt, 38

jcamvt (jcamst), 35

jcamvt.predict (jcamst.predict), 37

JCM.template (do.template), 18

Lympho, 39

msmsn (msmvt), 39

msmst (msmvt), 39

msmvn, 36

msmvn (msmvt), 39

msmvt, 36, 39

mvt.dof, 41

plotmixt.skew, 42

rdemmix, 3, 7–10, 22, 26, 45

`rdemmix2`, 26
`rdemmix2`(`rdemmix`), 45
`rdemmix3`(`rdemmix`), 45
`rdmsn`, 8–10, 26, 46
`rdmsn`(`ddmsn`), 7
`rdmst`, 7, 9, 10, 26, 46
`rdmst`(`ddmst`), 8
`rdmvn`, 7, 8, 10, 26, 46
`rdmvn`(`ddmvn`), 9
`rdmvt`, 7–9, 26, 46
`rdmvt`(`ddmvt`), 10