Package 'MixfMRI'

April 26, 2018

Version 0.1-0

Date 2018-04-25

Title Mixture fMRI Clustering Analysis

Depends R (>= 3.4.0)

Imports MASS, Matrix, RColorBrewer, fftw, MixSim, EMCluster

Enhances pbdMPI (>= 0.3-4), AnalyzeFMRI, oro.nifti

LazyLoad yes

LazyData yes

Description Utilizing model-based clustering (unsupervised)
for functional magnetic resonance imaging (fMRI) data.
The developed methods (Chen and Maitra (2018, manuscript)) include
2D and 3D clustering analyses (for p-values with voxel locations) and
segmentation analyses (for p-values alone) for fMRI data where p-values
indicate significant level of activation responding to stimulate
of interesting. The analyses are mainly identifying active
voxel/signal associated with normal brain behaviors.
Analysis pipelines (R scripts) utilizing this package
(see examples in 'inst/workflow/') is also implemented with high
performance techniques.

License Mozilla Public License 2.0

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

URL https://github.com/snoweye/MixfMRI

NeedsCompilation yes

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

Author Wei-Chen Chen [aut, cre],
Ranjan Maitra [aut],
Dan Nettleton [ctb]

Repository CRAN

Date/Publication 2018-04-26 11:38:14 UTC

R topics documented:

`MixfMRI-package` *fMRI Clustering Analysis*

Description

Utilizing model-based clustering (unsupervised) for fMRI data especially in a distributed manner. The methods includes 2D and 3D clustering analyses and segmentation analyses for fMRI signals where p-values are significant levels of active voxels which respond to stimulate of interesting. The analyses are mainly identifying active voxels/signals from normal brain behaviors. Workflows are also implemented utilizing high performance techniques.

Details

Package:	MixfMRI
Type:	Package
License:	GPL (>= 2)
LazyLoad:	yes

The main function of this package is `fclust()` that implements model-based clustering algorithm for fMRI signal data and provides unsupervised clustering results for the data. Several workflows implemented with high-performance computing techniques are also built in for automatically pro-

cess clustering, hypothesis, cluster merging, and visualizations.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

fclust(), set.global().

Examples

```
library(MixfMRI, quietly = TRUE)

  demo(fclust3d,'MixfMRI',ask=FALSE,echo=FALSE)
  demo(fclust2d,'MixfMRI',ask=FALSE,echo=FALSE)
```

---

algorithm *Main algorithms implemented in fclust*

Description

Main algorithms implemented in fclust.

Usage

```
ecm.step.gbd(PARAM.org)

apecma.step.gbd(PARAM.org)

em.step.gbd(PARAM.org)
```

Arguments

PARAM.org	an initialized PARAM, usually returned by set.global(), initial.em.gbd(), and initial.RndEM.gbd().

Details

These are main algorithms implemented in fclust().

Value

Return an optimized PARAM.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

set.global(), fclust(), PARAM, PARAM.org.

Examples

```
library(MixfMRI, quietly = TRUE)
library(EMCluster, quietly = TRUE)
# .FC.CT$algorithm <- "em"
# .FC.CT$model.X <- "V"
# .FC.CT$ignore.X <- TRUE
.FC.CT$check.X.unit <- FALSE

### Test toy1.
set.seed(1234)
X.gbd <- toy1$X.gbd
PV.gbd <- toy1$PV.gbd
PARAM <- set.global(X.gbd, PV.gbd, K = 2)
PARAM.new <- initial.em.gbd(PARAM)
PARAM.toy1 <- em.step.gbd(PARAM.new)
id.toy1 <- .MixfMRIEnv$CLASS.gbd
print(PARAM.toy1$ETA)
RRand(toy1$CLASS.gbd, id.toy1)


  ### Test toy2.
  set.seed(1234)
  X.gbd <- toy2$X.gbd
  PV.gbd <- toy2$PV.gbd
  PARAM <- set.global(X.gbd, PV.gbd, K = 3)
  PARAM.new <- initial.em.gbd(PARAM)
  PARAM.toy2 <- em.step.gbd(PARAM.new)
  id.toy2 <- .MixfMRIEnv$CLASS.gbd
  print(PARAM.toy2$ETA)
  RRand(toy2$CLASS.gbd, id.toy2)
```

Compute Q values *Q-values using Benjamini and Hochberg (1995)*

Description

Compute q-values Benjamini and Hochberg's (1995) approach for controlling FDR.

Usage

```
qvalue(p, method = c("BH1995", "BY2001"))
```

Arguments

p	a p-value vector.
method	using method by either BH1995 or BY2001

Details

This function compute q-values using Benjamini and Hochberg's (1995) approach for controlling FDR. The function `bh.fdr` is originally written by Dr. Dan Nettleton.

The Benjamini and Yeekutieli's (2001) approach for controlling FDR using the function `by.fdr` is coded by Wei-Chen Chen.

Value

Return corresponding q-values for the input p-values.

Author(s)

Dan Nettleton.

Modified by Wei-Chen Chen.

References

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

See Also

`dpval()`, `dmixpval()`.

Examples

```
library(MixfMRI, quietly = TRUE)
set.seed(1234)
da <- gendataset(phantom = shepp1fMRI, overlap = 0.01)
p <- da$pval[!is.na(da$pval)][1:100]
qvalue(p)
```

Compute Statistics for Log Odds Ratio of Posterior Probability

Compute Statistics for Log Odds Ratio of Posterior Probability

Description

The function computes statistics for log odds ratio of posterior probability.

Usage

```
logor.stat(x, fcobj, post.z, cov.param = NULL, cov.post.z = NULL,
               cov.logit.z = NULL, all.x = FALSE, drop.ETA1 = FALSE)
```

Arguments

`x`	an input list of two elements `X.gbd` and `PV.gbd`.
`fcobj`	a `fclust` object.
`post.z`	a matrix of `dim = N * K` for posterior probabilities, which is also the return value of `post.prob()`.
`cov.param`	a covariance matrix of `dim = d * d` for parameters, which is also a return of `cov.param()`. `d` is total number of parameters which is dependent on data and models.
`cov.post.z`	a covariance list of length equal to number of active voxels, which is also a return of `cov.post.z()`.
`cov.logit.z`	a covariance list of length equal to number of active voxels, which is also a return of `cov.logit.z()`.
`all.x`	all cov matrices for all observations are returned if TRUE, while for only active observations (those of class ids are greater than 1) if FALSE.
`drop.ETA1`	if drop the `ETA[1]` from the cov matrix due to the `min.1st.prop` constrain.

Details

For posterior probability, this function compute log odd ratio, cov matrix of log odd ratio, degrees of freedom, and testing statistics.

Value

A list is returned with four elements: `log.or`, `cov.log.or`, `df`, and `test.stat`.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

post.prob(), cov.param(), cov.post.z(), cov.logit.z().

Examples

```
library(MixfMRI, quietly = TRUE)
.FC.CT$model.X <- "I"
.FC.CT$CONTROL$debug <- 0
K <- 3

### Fit toy1.
set.seed(1234)
X.gbd <- toy1$X.gbd
X.range <- apply(X.gbd, 2, range)
X.gbd <- t((t(X.gbd) - X.range[1,]) / (X.range[2,] - X.range[1,]))
PV.gbd <- toy1$PV.gbd
fcobj <- fclust(X.gbd, PV.gbd, K = K, min.1st.prop = 0.5)

### Test log odds ratio.
x <- list(X.gbd = X.gbd, PV.gbd = PV.gbd)
post.z <- post.prob(x, fcobj)
lor <- logor.stat(x, fcobj, post.z)

### Check if 95% CE covers log odd ratio = 1.
id <- !is.na(lor$df)
id.cover.0 <- which(lor$test.stat[id] < pchisq(0.95, lor$df[id]))

### Get voxels needed for merging.
id.active <- which(fcobj$class != 1)
id.merge <- id.active[id][id.cover.0]

### Check results.
post.z[id.merge,]
cbind(toy1$X.gbd[id.merge,], toy1$PV.gbd[id.merge])
```

Covariance Matrices *Covariance Matrices*

Description

These functions compute posterior probabilities, Fisher information with covariance matrix of parameters, covariance matrix of posterior probabilities, and covariance matrix of logit posterior probabilities.

Usage

```
post.prob(x, fcobj)
cov.param(x, fcobj, post.z, drop.ETA1 = FALSE)
cov.post.z(x, fcobj, post.z, cov.param = NULL, all.x = FALSE,
```

```
                drop.ETA1 = FALSE)
cov.logit.z(x, fcobj, post.z, cov.param = NULL, cov.post.z = NULL,
                all.x = FALSE, drop.ETA1 = FALSE)
```

Arguments

x	an input list of two elements X.gbd and PV.gbd.
fcobj	a fclust object.
post.z	a matrix of dim = N * K for posterior probabilities, which is also the return value of post.prob().
cov.param	a covariance matrix of dim = d * d for parameters, which is also a return of cov.param(). d is total number of parameters which is dependent on data and models.
cov.post.z	a covariance list of length equal to number of active voxels, which is also a return of cov.post.z().
all.x	all cov matrices for all observations are returned if TRUE, while for only active observations (those of class ids are greater than 1) if FALSE.
drop.ETA1	if drop the ETA[1] from the cov matrix due to the min.1st.prop constrain.

Details

These functions are required to compute covariance matrices of parameters and posterior probabilities.

Use post.prob() to get the posterior probabilities.

Input the returns of post.prob() to cov.param() to obtain the cov matrix for parameters (inversed Fisher information obtained from inner product of gradient of log observed data likelihood). A list is returned with I for Fisher information, and cov for the covariance matrix which is inverted by ginv().

Input the returns of post.prob() and cov.param() to cov.post.z() to obtain the cov matrix for posterior probabilities by the multivariate delta method on the cov matrix for parameters.

Input the returns of post.prob(), cov.param(), and cov.post.z() to cov.logit.z() to obtain cov matrix for logit posterior probabilities by the multivariate delta method on cov matrix of posterior probabilities.

Value

A matrix or a list is returned.

The cov.param() will return a list containing two elements I for the Fisher information, and cov for the covariance matrix by generalized inversed of the Fisher information. The dimension of both elements are d * d where d = K * 7 - 4 for 2D data and d = K * 9 - 4 for 3D data if drop.ETA1 = TRUE, otherwise they are d = K * 7 - 3 and d = K * 9 -4, respectively.

The cov.post.z() will return a list containing cov matrices of posterior probabilities for each valid/selected voxel.

The cov.logit.z() will return a list containing cov matrices of logit posterior probabilities for each valid/selected voxel.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

EMCluster::lmt(), lmt.I().

Examples

```
library(MixfMRI, quietly = TRUE)
library(EMCluster, quietly = TRUE)
.FC.CT$model.X <- "I"
.FC.CT$CONTROL$debug <- 0
K <- 3


  ### Fit toy1.
  set.seed(1234)
  X.gbd <- toy1$X.gbd
  X.range <- apply(X.gbd, 2, range)
  X.gbd <- t((t(X.gbd) - X.range[1,]) / (X.range[2,] - X.range[1,]))
  PV.gbd <- toy1$PV.gbd
  fcobj <- fclust(X.gbd, PV.gbd, K = K, min.1st.prop = 0.5)

  ### Test cov matrix of posterior z and logit posterior z.
  x <- list(X.gbd = X.gbd, PV.gbd = PV.gbd)
  post.z <- post.prob(x, fcobj)
  cov.param <- cov.param(x, fcobj, post.z = post.z)
  cov.post.z <- cov.post.z(x, fcobj, post.z = post.z,
                           cov.param = cov.param$cov)
  cov.logit.z <- cov.logit.z(x, fcobj, post.z = post.z,
                             cov.param = cov.param$cov,
                             cov.post.z = cov.post.z)

  ### Compute cov matrix of log odds ratio for all k > 1.
  A <- cbind(rep(-1, K - 1), diag(1, K - 1))
  logit.p <- log(post.z[fcobj$class != 1,] / (1 - post.z[fcobj$class != 1,]))
  log.or <- logit.p %*% t(A)
  cov.log.or <- lapply(cov.logit.z, function(x) A %*% x %*% t(A))

  ### Check if 0 vector covered by 95% confidence ellipsoid.
  id <- 1
  plot(log.or[id,],
       xlim = log.or[id, 1] + c(-5, 5),
       ylim = log.or[id, 2] + c(-5, 5),
       main = "1st observation", xlab = "x", ylab = "y")
  plotBN(log.or[id,], cov.log.or[[id]])
  points(0, 0, col = 2)
```

Covariance Matrices of Logit ETA

Covariance Matrices of Logit ETA

Description

These functions computes covariance matrix of logit ETA.

Usage

```
cov.logit.ETA(x, fcobj, cov.param = NULL)
```

Arguments

`x`	an input list of two elements `X.gbd` and `PV.gbd`.
`fcobj`	a `fclust` object.
`cov.param`	a covariance matrix of `dim = d * d` for parameters, which is also a return of `cov.param()`. `d` is total number of parameters which is dependent on data and models.

Details

These functions are required to compute covariance matrices of logit ETA.

Input the returns of `cov.param()` to `cov.logit.ETA()` to obtain the cov matrix for logit ETA by the multivariate delta method on the cov matrix for parameters.

Value

A matrix.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

`EMCluster::lmt()`, `lmt.I()`.

Examples

```
library(MixfMRI, quietly = TRUE)
.FC.CT$model.X <- "I"
.FC.CT$CONTROL$debug <- 0
K <- 3


  ### Fit toy1.
  set.seed(1234)
  X.gbd <- toy1$X.gbd
  X.range <- apply(X.gbd, 2, range)
  X.gbd <- t((t(X.gbd) - X.range[1,]) / (X.range[2,] - X.range[1,]))
  PV.gbd <- toy1$PV.gbd
  fcobj <- fclust(X.gbd, PV.gbd, K = K, min.1st.prop = 0.5)

  ### Test cov matrix of posterior z.
  x <- list(X.gbd = X.gbd, PV.gbd = PV.gbd)
  post.z <- post.prob(x, fcobj)
  cov.param <- cov.param(x, fcobj, post.z)
  cov.logit.ETA <- cov.logit.ETA(x, fcobj, cov.param = cov.param$cov)

  ### Compute cov matrxi of eta_k - eta_1 for all k > 1.
  A <- cbind(rep(-1, K - 1), diag(1, K - 1))
  ETA <- fcobj$param$ETA
  log.or <- log(ETA / (1 - ETA)) %*% t(A)
  cov.log.or <- A %*% cov.logit.ETA %*% t(A)
```

`Density function of p-values`

Density function of p-values

Description

These functions based on normal assumption and transformation to derive a (mixture) density function of p-values.

Usage

```
dpval(x, mu = 0, log = FALSE)
dmixpval(x, eta, mu)
```

Arguments

`x`	support of p-values which should be between 0 and 1.
`mu`	hypothetical mean of testing statistics (in normal distribution) for producing p-values.
`log`	if return log of density.
`eta`	mixing proportion of `K` components if a mixture is assumed.

Details

Note that `eta` and `mu` in `dmixpval()` are of length `K` for `K` component mixtures.

Value

Corresponding density values (to the input `x`) are returned.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

`gendataset()`, `qvalue()`.

Examples

```
library(MixfMRI, quietly = TRUE)
set.seed(1234)
da <- gendataset(phantom = shepp1fMRI, overlap = 0.01)
x <- da$pval[!is.na(da$pval)][1:100]
dpval(x)
dmixpval(x, mu = da$mu, eta = da$eta)
```

`Example Datasets` *Example datasets in MixfMRI*

Description

These are datasets used to demo examples and workflows in this package.

Format

Objects may contain several information and data.

Details

`pstats` is a 3D example.

`pval.2d.complex` and `pval.2d.mag` are 2D examples.

`shepp0fMRI`, `shepp1fMRI`, `shepp2fMRI` and `sheppAnat` are phantoms generated by Dr. Maitra for simulation studies with different overlap levels for p-values.

`toy*` are 2D toy examples.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

Examples

```
library(MixfMRI, quietly = TRUE)

### Plotting.
demo(shepp,'MixfMRI',ask=FALSE,echo=FALSE)
```

False Discovery Rates for Spatial Signals

False Discovery Rates for Spatial Signals using Benjamini and Heller (2007)

Description

Compute q-values Benjamini and Heller's (2007) approach for controlling FDR for spatial signals.

Usage

```
fdr.bh.p1(p, w = rep(1, length(p)), q = 0.05)
fdr.bh.p2(p, w = rep(1, length(p)), q = 0.05)
```

Arguments

p	a p-value vector. No NA is allowed and all values are in [0, 1].
w	a weight vector for p-values.
q	a desired cutoff for adjusting p-values.

Details

These functions implement first two procedures in Benjamini and Heller (2007) for controlling FDR for spatial signals.

Value

Return the number of rejected hypotheses and all corresponding q-values for the input p-values.

Author(s)

Wei-Chen Chen.

References

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

See Also

qvalue().

Examples

```
library(MixfMRI, quietly = TRUE)
set.seed(1234)
da <- gendataset(phantom = shepp1fMRI, overlap = 0.01)
p <- da$pval[!is.na(da$pval)][1:100]
fdr.bh.p1(p)
fdr.bh.p2(p)
```

Generalized Cluster-Based Analysis (CBA) Method

Generalized Cluster-Based Analysis (CBA) Method

Description

Find clusters in 2D or 3D based on a generalized CBA method. The CBA method is originally proposed by Heller, et.al. (2006) using the correlation of two time series as the similarity of two spatial locations.

Usage

```
cba.cor(da.ts, da.m = NULL, adj.dist = TRUE, fun.sim = stats::cor)
cba.cor.2d(da.ts, da.m = NULL, adj.dist = TRUE, fun.sim = stats::cor)
cba.cor.3d(da.ts, da.m = NULL, adj.dist = TRUE, fun.sim = stats::cor)
```

Arguments

da.ts	a time series array of dimensions x * y * z * t.
da.m	a mask determining inside of brain or not.
adj.dist	if adjust correlations by distance.
fun.sim	a function computing simility of two locations.

Details

These functions implement the 2D and 3D versions of CBA proposed by Heller, et.al. (2006).

da.ts should have dimensions x * y * z * t for 3D data and x * y * time for 2D data. Similarly, da.m would have x * y * z and x * y correspondingly.

da.m has values 0 or 1 indicating outside or inside a brain, respectively.

`fun.sim(a, B)` is a function return similarity between a location `a` and N neighboring locations `B` where `a` is of dimension `t * 1` and `B` is of dimensiont * N. Ideally, `fun.sim()` should return values of similarity which take values between 0 and 1 where 0 means totally different and 1 means completely identical of two spatial locations. By default, `stats::cor` is used. See the example section next for user defined functions for `fun.sim()`.

Value

Return the cluster ids for each voxel. NA for outside of brain if `da.m` is provided.

Author(s)

Wei-Chen Chen.

References

Heller, et.al. (2006) "Cluster-based analysis of FMRI data", NeuroImage, 33(2), 599-608.

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

See Also

`fdr.bh.p1()`, `fdr.bh.p2()`.

Examples

```
### Simulated data
library(MixfMRI, quietly = TRUE)
dim <- c(4, 5, 4, 10)
set.seed(123)
da.ts <- array(rnorm(prod(dim)), dim = dim)
id.class <- suppressWarnings(cba.cor(da.ts))
table(id.class)

fun.tanh <- function(a, B){
  d <- 1 / apply(B, 2, function(b){ dist(rbind(as.vector(a), b)) })
  tanh(d)
}
id.class.tanh <- suppressWarnings(cba.cor(da.ts, fun.sim = fun.tanh))
table(id.class.tanh)

fun.logit <- function(a, B){
  d <- dist(t(cbind(a, B)))[1:ncol(B)]
  (1 / (1 + exp(-d))) * 2 - 1
}
id.class.logit <- suppressWarnings(cba.cor(da.ts, fun.sim = fun.logit))
table(id.class.logit)


  ### Real data
  library(AnalyzeFMRI, quietly = TRUE)
  library(oro.nifti, quietly = TRUE)
```

```
fn <- "pb02_volreg_tlrc.nii"
da <- readNIfTI(fn)
da.ts <- da@.Data

fn <- "mask_anat.nii"
da <- readNIfTI(fn)
da.m <- da@.Data

id.class <- suppressWarnings(cba.cor(da.ts, da.m))
dim(id.class) <- dim(da.m)
length(table(id.class))
```

`initial` *Main initialization functions*

Description

Main initialization functions.

Usage

```
initial.em.gbd(PARAM)

initial.RndEM.gbd(PARAM)
```

Arguments

PARAM	a list of uninitialized parameters, as usual, the returned values of `set.global()`, to be initialized according to data (inside PARAM).

Details

`initial.em.gbd()` takes in a template of PARAM (uninitialized), and usually is available by calling `set.global()`, then return an initialized PARAM which is ready for EM runs.

Internally, there are six different initializations implemented for the function `initial.em.gbd()` including `prob.extend`, `prob.simple`, `qnorm.extend`, `qnorm.simple`, `extend`, and `simple`. These methods are mainly based on transformation of original space of data (p-values and voxel locations) into more linear space such that the Euclidean distance more makes sense (fairly) to classify data in groups.

`initial.RndEM.gbd()` implement RndEM initialization algorithm based on repeated calling `initial.em.gbd()`.

Note that all configurations are included in PARAM set by `set.global()`.

Value

These functions return an initialized PARAM for EM runs based on pre-stored configuration within the input uninitialized PARAM.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

set.global(), fclust(), PARAM.

Examples

```
library(MixfMRI, quietly = TRUE)
library(EMCluster, quietly = TRUE)
# .FC.CT$algorithm <- "em"
# .FC.CT$model.X <- "V"
# .FC.CT$ignore.X <- TRUE
.FC.CT$check.X.unit <- FALSE

### Test toy1.
set.seed(1234)
X.gbd <- toy1$X.gbd
PV.gbd <- toy1$PV.gbd
PARAM <- set.global(X.gbd, PV.gbd, K = 2)
PARAM.new <- initial.em.gbd(PARAM)
PARAM.toy1 <- em.step.gbd(PARAM.new)
id.toy1 <- .MixfMRIEnv$CLASS.gbd
print(PARAM.toy1$ETA)
RRand(toy1$CLASS.gbd, id.toy1)


  ### Test toy2.
  set.seed(1234)
  X.gbd <- toy2$X.gbd
  PV.gbd <- toy2$PV.gbd
  PARAM <- set.global(X.gbd, PV.gbd, K = 3)
  PARAM.new <- initial.em.gbd(PARAM)
  PARAM.toy2 <- em.step.gbd(PARAM.new)
  id.toy2 <- .MixfMRIEnv$CLASS.gbd
  print(PARAM.toy2$ETA)
  RRand(toy2$CLASS.gbd, id.toy2)
```

Description

These functions test two mixture Gaussian fMRI models with identity covariance matrices and different numbers of clusters. These functions are similar to the `EMCluster::lmt()`, but is coded for fMRI models in **MixfMRI**.

Usage

```
lmt.I(fcobj.0, fcobj.a, X.gbd, PV.gbd, tau = 0.5, n.mc.E.delta = 1000,
      n.mc.E.chi2 = 1000, verbose = FALSE)
lmt.pv(fcobj.0, fcobj.a, X.gbd, PV.gbd, tau = 0.5, n.mc.E.delta = 1000,
      n.mc.E.chi2 = 1000, verbose = FALSE)
```

Arguments

`fcobj.0`	a `fclust` object for the null hypothesis.
`fcobj.a`	a `fclust` object for the alternative hypothesis.
`X.gbd`	a data matrix of `N` voxel locations. `dim(X.gbd) = N x 3` for 3D data and `N x 2` for 2D data.
`PV.gbd`	a p-value vector of signals associated with voxels. `length(PV.gbd) = N`.
`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 H0: `fcobj.0` v.s. Ha: `fcobj.a`.

`lmt.pv()` only uses `PV.gbd`.

Value

A list of class `lmt.I` are returned.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

`EMCluster::lmt()`.

Examples

```
library(MixfMRI, quietly = TRUE)
library(EMCluster, quietly = TRUE)
.FC.CT$model.X <- "I"
.FC.CT$check.X.unit <- FALSE
.FC.CT$CONTROL$debug <- 0


  ### Fit toy1.
  set.seed(1234)
  X.gbd <- toy1$X.gbd
  PV.gbd <- toy1$PV.gbd
  ret.2 <- fclust(X.gbd, PV.gbd, K = 2)
  ret.3 <- fclust(X.gbd, PV.gbd, K = 3)
  ret.4 <- fclust(X.gbd, PV.gbd, K = 4)
  ret.5 <- fclust(X.gbd, PV.gbd, K = 5)

  ### ARI
  RRand(toy1$CLASS.gbd, ret.2$class)
  RRand(toy1$CLASS.gbd, ret.3$class)
  RRand(toy1$CLASS.gbd, ret.4$class)
  RRand(toy1$CLASS.gbd, ret.5$class)

  ### Test toy1.
  (lmt.23 <- lmt.I(ret.2, ret.3, X.gbd, PV.gbd))
  (lmt.24 <- lmt.I(ret.2, ret.4, X.gbd, PV.gbd))
  (lmt.25 <- lmt.I(ret.2, ret.5, X.gbd, PV.gbd))
  (lmt.34 <- lmt.I(ret.3, ret.4, X.gbd, PV.gbd))
  (lmt.35 <- lmt.I(ret.3, ret.5, X.gbd, PV.gbd))
  (lmt.45 <- lmt.I(ret.4, ret.5, X.gbd, PV.gbd))

  ### Test toy1 using p-values only.
  (lmt.pv.23 <- lmt.pv(ret.2, ret.3, X.gbd, PV.gbd))
  (lmt.pv.24 <- lmt.pv(ret.2, ret.4, X.gbd, PV.gbd))
  (lmt.pv.25 <- lmt.pv(ret.2, ret.5, X.gbd, PV.gbd))
  (lmt.pv.34 <- lmt.pv(ret.3, ret.4, X.gbd, PV.gbd))
  (lmt.pv.35 <- lmt.pv(ret.3, ret.5, X.gbd, PV.gbd))
  (lmt.pv.45 <- lmt.pv(ret.4, ret.5, X.gbd, PV.gbd))
```

LRT *Likelihood ratio tests*

Description

Likelihood ratio tests for merging clusters.

Usage

```
lrt(PV.gbd, CLASS.gbd, K, H0.alpha = .FC.CT$LRT$H0.alpha,
    H0.beta = .FC.CT$LRT$H0.beta)

lrt2(PV.gbd, CLASS.gbd, K, H0.mean = .FC.CT$LRT$H0.mean,
  upper.beta = .FC.CT$INIT$BETA.beta.max, proc = c("1", "2", "weight"))

lrt.betamean(PV.gbd, CLASS.gbd, K, proc = c("1", "2"))

lrt.betaab(PV.gbd, CLASS.gbd, K, proc = c("1", "2"))
```

Arguments

PV.gbd	a p-value vector of signals associated with voxels. length(PV.gbd) = N.
CLASS.gbd	a classification vector of signals associated with voxels. length(CLASS.gbd) = N.
K	number of clusters.
H0.alpha	null hypothesis for the alpha parameter of Beta distribution.
H0.beta	null hypothesis for the beta parameter of Beta distribution.
H0.mean	null hypothesis for the mean of Beta distribution.
upper.beta	BETA.beta.max, maximum value of beta parameter of Beta distribution.
proc	q-value procedure for adjusting p-values.

Details

These functions perform likelihood ratio tests for merging clusters. Only p-values coordinates (Beta density) are tested, while voxel location coordinates (multivariate Normal density) are not involved in testing.

lrt.betamean tests if means of any two pairs of mixture (p-value) component were the same. The chi-square distribution with 1 degree of freedom is used.

lrt.betaab tests if alpha and beta of any two pairs of mixture (p-value) components were the same. The chi-square distribution with 2 degrees of freedom is used.

Value

A matrix contains MLEs of parameters of Beta distribution under the null hypothesis and the union of null and alternative hypotheses. The matrix also contains testing statistics and p-values.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

PARAM.

Examples

```
library(MixfMRI, quietly = TRUE)
set.seed(1234)

### Test 2d data.
da <- pval.2d.mag
id <- !is.na(da)
PV.gbd <- da[id]
id.loc <- which(id, arr.ind = TRUE)
X.gbd <- t(t(id.loc) / dim(da))
ret <- fclust(X.gbd, PV.gbd, K = 2, min.1st.prop = 0.95)
# print(ret)

### p-values of rest clusters.
ret.lrt <- lrt(PV.gbd, ret$class, K = 2)
print(ret.lrt)

  ret.lrt2 <- lrt2(PV.gbd, ret$class, K = 3)
  print(ret.lrt2)
```

Main functions *Main MixfMRI function*

Description

Main MixfMRI functions.

Usage

```
fclust(X.gbd, PV.gbd, K = 2,
  PARAM.init = NULL,
  min.1st.prop = .FC.CT$INIT$min.1st.prop,
  max.PV = .FC.CT$INIT$max.PV,
  class.method = .FC.CT$INIT$class.method[1],
  RndEM.iter = .FC.CT$CONTROL$RndEM.iter,
  algorithm = .FC.CT$algorithm[1],
  model.X = .FC.CT$model.X[1],
  ignore.X = .FC.CT$ignore.X,
  stop.unstable = TRUE,
  MPI.gbd = .FC.CT$MPI.gbd, common.gbd = .FC.CT$common.gbd)

set.global(X.gbd, PV.gbd, K = 2,
  min.1st.prop = .FC.CT$INIT$min.1st.prop,
```

```
max.PV = .FC.CT$INIT$max.PV,
class.method = .FC.CT$INIT$class.method[1],
RndEM.iter = .FC.CT$CONTROL$RndEM.iter,
algorithm = .FC.CT$algorithm[1],
model.X = .FC.CT$model.X[1],
ignore.X = .FC.CT$ignore.X,
check.X.unit = .FC.CT$check.X.unit,
MPI.gbd = .FC.CT$MPI.gbd, common.gbd = .FC.CT$common.gbd)
```

Arguments

X.gbd	a data matrix of N voxel locations. dim(X.gbd) = N x 3 for 3D data and N x 2 for 2D data.
PV.gbd	a p-value vector of signals associated with voxels. length(PV.gbd) = N.
K	number of clusters to be estimated.
PARAM.init	initial parameters.
min.1st.prop	lower bound of mixing proportion (ETA) of the 1st cluster (uniform).
max.PV	upper bound of p-values where initializations pick from.
class.method	classification method for initializations.
RndEM.iter	number of RndEM iterations.
algorithm	either "ecm" (ECM), "apecma" (APECMa) or "em" (EM) algorithm.
model.X	either "I" or "V" for covariance matrix.
ignore.X	if X.gbd used in model, TRUE for PV.gbd only.
check.X.unit	if X.gbd are all in [0, 1].
stop.unstable	if fclust stops if unstable results occur.
MPI.gbd	if MPI ("EGM" algorithm) is used.
common.gbd	if X.gbd and PV.gbd are in common across all ranks when MPI.gbd = TRUE.

Details

The fclust() contains initialization and EM algorithms for clustering fMRI signal data which have two parts: X.gbd for voxel information either 2D or 3D, PV.gbd for p-value of signals associated with voxels. Each signal is assumed as a mixture distribution with K components with mixing proportion ETA, and each component has two independent coordinates with density functions: Beta and multivariate Normal distributions.

Beta density: The 1st component is restricted by min.1st.prop and Beta(1, 1) distribution. The other K - 1 components have Beta(alpha, beta) distribution with alpha < 1 < beta.

Multivariate Normal density: model.X = "I" is for identity cov matrix of multivariate Normal distribution, and "V" for unstructured cov matrix. ignore.X = TRUE is to ignore X.gbd and normal density, i.e. only Beta density is used.

Currently, APECMa and EM algorithms are implemented with EGM algorithm to speed up convergence if MPI is available. RndEM initialization is also implemented for better chance of good initial values for convergence.

The set.global() has purposes: create a template/storage of parameters, save configurations, and called by fclust() to initial the parameters, such as initial.em.gbd() or initial.RndEM.gbd().

Value

A list with class `fclust` by `fclust()` is returned which can be summarized by `print.fclust()`.

A list `PARAM` or `PARAM.org` is returned by `set.global()`:

`N.gbd`	number of observations (within the rank), and should be equal to `N.all` if `MPI.gbd = FALSE`.
`N.all`	numbers of observations (of all ranks if `MPI.gbd = TRUE`).
`N`	total number of observations (`sum(N.all)`).
`p`	dimension of an observation (3 for 2D signals, 4 for 3D signals), equivalent to total number of coordinates.
`p.X`	dimension of `X.gbd` (2 for 2D signals, 3 for 3D signals, 0 when `ignore.X = TRUE`, number of voxel coordinates.
`K`	number of clusters.
`ETA`	mixing proportion, length `K`.
`log.ETA`	`log(ETA)`.
`BETA`	a list of length `K` containing parameters (alpha, beta) of Beta density.
`MU`	a matrix of dimension `p.X by K`.
`SIGMA`	a list of length `K`, and each is of dimension `K x K`.
`logL`	log likelihood value.
`min.1st.prop`	carried from input.
`max.PV`	carried from input.
`class.method`	classification method of initializations.
`min.N.CLASS`	`p + 1`.
`model.X`	carried from input.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

`print.fclust()`.

Examples

```
library(MixfMRI, quietly = TRUE)
library(EMCluster, quietly = TRUE)
# .FC.CT$algorithm <- "em"
# .FC.CT$model.X <- "V"
# .FC.CT$ignore.X <- TRUE
.FC.CT$check.X.unit <- FALSE
set.seed(1234)

### Test toy1.
X.gbd <- toy1$X.gbd[, -3]
PV.gbd <- toy1$PV.gbd
PARAM <- fclust(X.gbd, PV.gbd, K = 2)
print(PARAM)
id.toy1 <- .MixfMRIEnv$CLASS.gbd
print(RRand(toy1$CLASS.gbd, id.toy1))


  ### Test toy2.
  X.gbd <- toy2$X.gbd[, -3]
  PV.gbd <- toy2$PV.gbd
  PARAM <- fclust(X.gbd, PV.gbd, K = 3)
  print(PARAM)
  id.toy2 <- .MixfMRIEnv$CLASS.gbd
  print(RRand(toy2$CLASS.gbd, id.toy2))
```

`MixfMRI Control` *Sets of controls in MixfMRI*

Description

These sets of controls are used to provide default values in this package.

Format

Objects contain several parameters for methods.

Details

The elements of `.FC.CT` are default values for main controls of **MixfMRI** including

Elements	Default	Usage
`algorithm`	"apecma"	implemented algorithm
`optim.method`	"BFGS"	optimization method
`model.X`	"I"	cov matrix structure
`ignore.X`	FALSE	if using voxel information
`check.X.unit`	TRUE	if checking X in `[0, 1]`
`CONTROL`	a list	see CONTROL next for details

INIT	a list	see INIT next for details
LRT	a list	see LRT next for details
MPI.gbd	FALSE	if MPI speedup available
common.gbd	TRUE	if X in common gbd format

The elements of CONTROL are default values for optimization controls of implemented EM algorithm including

Elements	Default	Usage
max.iter	1000	maximum number of EM iterations
abs.err	1e-4	absolute error of convergence
rel.err	1e-6	relative error of convergence
debug	1	debugging level
RndEM.iter	10	RndEM iterations
exp.min	log(.Machine$double.xmin)	minimum exponential power
exp.max	log(.Machine$double.xmax)	maximum exponential power
sigma.ill	1e-6	ill condition limit
DS.max	1e+4	maximum chol() cov matrix
DS.min	1e-6	minimum chol() cov matrix

The elements of INIT are default values or limitations for initial parameters implemented for EM algorithm including

Elements	Default	Usage
min.1st.prop	0.8	minimum proportion of 1st cluster
max.PV	0.1	maximum p-value for initialization
BETA.alpha.min	0 + 1e-6	minimum value of alpha parameter of Beta distribution
BETA.alpha.max	1 - 1e-6	maximum value of alpha parameter of Beta distribution
BETA.beta.min	1 + 1e-6	minimum value of beta parameter of Beta distribution
BETA.beta.max	1e+6	maximum value of beta parameter of Beta distribution
max.try.iter	10	maximum retry iterations if result is unstable
class.method	"prob.extned"	classification method at initializations

The elements of LRT are default values or limitations for likelihood ratio tests including

Elements	Default	Usage
H0.alpha	1	null hypothesis alpha parameter of Beta distribution
H0.beta	1	null hypothesis beta parameter of Beta distribution
H0.mean	0.05	null hypothesis mean of Beta distribution

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

set.global(), fclust().

Plotting *Main plotting function*

Description

Main plotting function in **MixfMRI**.

Usage

```
plotfclust(da, posterior, main = NULL, xlim = NULL, ylim = NULL)
plotfclustpv(da, posterior, main = NULL, xlim = NULL, ylim = NULL)

plotpv(da, posterior, PARAM, zlim = c(0, 0.01), plot.mean = TRUE,
       xlab = "", ylab = "", main = NULL, xlim = NULL, ylim = NULL,
       col = my.YlOrRd(), ignore.bg = FALSE)
plotpvlegend(zlim = c(0, 0.01), n.level = 20, main = NULL,
             col = my.YlOrRd())
```

Arguments

da	a data set to be plotted.
posterior	a posterior data set to be plotted.
PARAM	a returning parameter object from fclust().
main	title of the plot.
xlim	limits of x-axis.
ylim	limits of y-axis.
zlim	limits of z-axis.
xlab	labels of x-axis.
ylab	labels of y-axis.
plot.mean	if plotting mean values of each cluster.
col	colors to be drawn.
ignore.bg	if ignoring the background.
n.level	number of levels to be plotted.

Details

These are example functions to plot results, simulations, and datasets.

Value

Return plots.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

set.global().

Examples

```
library(MixfMRI, quietly = TRUE)
set.seed(1234)


  ### Check 2d data.
  da <- pval.2d.complex
  id <- !is.na(da)
  PV.gbd <- da[id]
  hist(PV.gbd, nclass = 100, main = "p-value")

  ### Test 2d data.
  id.loc <- which(id, arr.ind = TRUE)
  X.gbd <- t(t(id.loc) / dim(da))
  ret <- fclust(X.gbd, PV.gbd, K = 3)
  print(ret)

  ### p-values of rest clusters.
  ret.lrt <- lrt(PV.gbd, ret$class, K = 3)
  print(ret.lrt)
  ret.lrt2 <- lrt2(PV.gbd, ret$class, K = 3)
  print(ret.lrt2)

  ### Plotting.
  par(mfrow = c(2, 2), mar = c(0, 0, 2, 0))
  plotpv(da, ret$posterior, ret$param,
         zlim = c(0.005, 0.008), main = "Mean of Beta Distribution")
  plotpv(da, ret$posterior, ret$param,
         plot.mean = FALSE, main = "p-value")
  par(mar = c(5.1, 4.1, 4.1, 2.1))
  plotpvlegend(zlim = c(0.005, 0.008), main = "Mean of Beta Distribution")
  plotpvlegend(zlim = c(0, 0.01), main = "p-value")
```

`Print Objects` *Print fclust related outputs*

Description

Print flcust related outputs.

Usage

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

Arguments

`x`	an object with the class attributes.
`...`	other arguments to the `print` function.

Details

`x` is the return result from `fclust()`.

Value

A summary of `fclust` object is printed.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

`set.global()`, `fclust()`.

Examples

```
library(MixfMRI, quietly = TRUE)
set.seed(1234)

### Check 2d data.
da <- pval.2d.complex
id <- !is.na(da)
PV.gbd <- da[id]
# hist(PV.gbd, nclass = 100, main = "p-value")

### Test 2d data.
```

```
id.loc <- which(id, arr.ind = TRUE)
X.gbd <- t(t(id.loc) / dim(da))
ret <- fclust(X.gbd, PV.gbd, K = 2)
print(ret)
```

`Simulations` *Generate datasets for MixfMRI simulations*

Description

Generate datasets for MixfMRI simulations

Usage

```
gendataset(phantom, overlap)
```

Arguments

`phantom`	a phantom dataset.
`overlap`	a desired overlap level.

Details

This is a function to generate simulated fMRI data based on the input `phantom` and the desired `overlap` level for the fMRI p-value.

Value

Return a list contains `eta` for mixing proportion, `overlap` for the desired level, `mu` for center of p-values, `class.id` for the true classifications where p-values belong to, `tval` for the testing statistics, and `pval` for the p-values of interesting in simulations.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References

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

See Also

`set.global()`.

Examples

```
library(MixfMRI, quietly = TRUE)
set.seed(1234)
da <- gendataset(phantom = shepp1fMRI, overlap = 0.01)$pval
da2 <- gendataset(phantom = shepp2fMRI, overlap = 0.01)$pval

par(mfrow = c(2, 2), mar = rep(0.05, 4))
image(shepp1fMRI[50:210, 50:210], axes = FALSE)
image(shepp2fMRI[50:210, 50:210], axes = FALSE)
image(da[50:210, 50:210], axes = FALSE)
image(da2[50:210, 50:210], axes = FALSE)
```

Smoothing *Generate datasets with smoothing for MixfMRI simulations*

Description

Generate datasets with smoothing for MixfMRI simulations

Usage

```
gcv.smooth2d(y, interval)
```

Arguments

`y`	a set of p-values in 2d phantom
`interval`	an interval for `optimize` function.

Details

The function is used to smooth for Dr. Maitra's 2d phantom simulation. The smoothing method is based on Garcia (2010), CSDA.

Value

Return a list containing two elements `im.smooth` and `par.val`.

Author(s)

Ranjan Maitra.

References

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

`Summarized Overlap` *Summarized Overlap*

Description

Compute summarized overlap on a given overlap (symmetric) matrix.

Usage

```
summarized.overlap(overlap.mat)
```

Arguments

`overlap.mat`	an overlap (symmetric) matrix.

Details

`overlap.mat` is a `p * p` matrix containing pair wised overlaps of `p` experiments. `overlap.mat` is assumed a symmetric matrix. This function returns a summarized overlap based on the input `overlap.mat` that charactorizes the overall overlap behavior of the `p` experiments.

Value

A single value is returned.

Author(s)

Ranjan Maitra.

References

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

Examples

```
library(MixfMRI, quietly = TRUE)
set.seed(1234)
p <- 10  # 10 experiments.
overlap.mat <- diag(1, p)
overlap.mat[lower.tri(overlap.mat)] <- runif(p * (p - 1) / 2)
overlap.mat[upper.tri(overlap.mat)] <- t(overlap.mat)[upper.tri(overlap.mat)]
summarized.overlap(overlap.mat)
```

Index