Package ‘MixfMRI’

September 4, 2023

Version 0.1-3
Date 2023-09-03
Title Mixture fMRI Clustering Analysis
Depends R (>= 4.0.0)
Imports MASS, Matrix, RColorBrewer, fftw, MixSim, EMCluster
Enhances pbdMPI (>= 0.3-4), 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 (2023) <doi:10.1002/hbm.26425>) 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 [aut, ctb], Pierre Lafaye De Micheaux [aut, ctb] (Threshold functions from AnalyzeFMRI), Jonathan L Marchini [aut, ctb] (Threshold functions from AnalyzeFMRI)
MixfMRI-package

Description

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

Author(s)
Wei-Chen Chen and Ranjan Maitra.

References

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

Examples
```r
library(MixfMRI, quietly = TRUE)
.rem <- function()

.demo <- function()
  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 \texttt{set.global()}, \texttt{initial.em.gbd()}, and \texttt{initial.RndEM.gbd}.

Details

These are main algorithms implemented in \texttt{fclust}().

Value

Return an optimized PARAM.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References


See Also

\texttt{set.global()}, \texttt{fclust()}, \texttt{PARAM}, \texttt{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)

.rem <- function(){

### Test toy2.
set.seed(1234)


cluster.threshold

cluster.threshold <- function(x, nmat = NULL, level.thr = 0.5, size.thr)
{
  # Calculate contiguous clusters of locations in a 3D array that are above some threshold and with
  # some minimum size.

  # Usage
  cluster.threshold(x, nmat = NULL, level.thr = 0.5, size.thr)

  # Arguments
  x A 3D array
  nmat A matrix with 3 columns specifying the neighbourhood system. Default is 6
        nearest neighbours in 3D.
  level.thr The level at which to threshold the array values. Default is 0.5 and is designed
              to cluster 0-1 arrays.
  size.thr The cluster size threshold.

  # Details
  Note: This function is directly copied from "AnalyzeFMRI".

  # Value
  Returns an array of the same size as x with a 1 at all locations which have a value above level.thr
  and are in a cluster of similar locations with size greater than size.thr.

  # Author(s)
  J. L. Marchini
Examples

```r
x <- array(0, dim = c(64, 64, 21))
x[10:20, 10:20, 1:5] <- 1
x[30:40, 30:40, 6:7] <- 1
x[50, 50, 8:9] <- 1

a <- cluster.threshold(x, size.thr = 400)
sum(x) # should be 849
sum(a) # should be 605
```

---

**Compute Q values**

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

Description

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

Usage

```r
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 Yekutieli’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

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)

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

Usage
logor.stat(x, fobj, 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.
fobj 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.
Compute Statistics for Log Odds Ratio of Posterior Probability

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


See Also

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

Examples

```r
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

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.
fobj
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.
Covariance Matrices

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 \( d \times d \) where \( d = K \times 7 - 4 \) for 2D data and \( d = K \times 9 - 4 \) for 3D data if `drop.ETA1 = TRUE`, otherwise they are \( d = K \times 7 - 3 \) and \( d = K \times 9 - 4 \), 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


See Also

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

Examples

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

.rem <- function(){
    ### 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.
```
Covariance Matrices of Logit ETA

### Description

These functions compute covariance matrix of logit ETA.

### Usage

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

### Arguments

- **x**: an input list of two elements `X.gbd` and `PV.gbd`.
- **fobj**: 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


See Also

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

Examples

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

.rem <- function()
{
### 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 matrixi 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)
Density function of p-values

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


See Also
gendataset(), qvalue().
Examples

```r
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)
```

### EC.3D

**Expected Euler Characteristic for a 3D Random Field**

**Description**

Calculates the Expected Euler Characteristic for a 3D Random Field thresholded at a level \( u \).

**Usage**

```r
EC.3D(u, sigma, voxdim = c(1, 1, 1), num.vox, type = c("Normal", "t"), df = NULL)
```

**Arguments**

- **u**: The threshold for the field.
- **sigma**: The spatial covariance matrix of the field.
- **voxdim**: The dimensions of the cuboid 'voxels' upon which the discretized field is observed.
- **num.vox**: The number of voxels that make up the field.
- **type**: The marginal distribution of the Random Field (only Normal and t at present).
- **df**: The degrees of freedom of the t field.

**Details**

The Euler Characteristic \( \chi_u \) (Adler, 1981) is a topological measure that essentially counts the number of isolated regions of the random field above the threshold \( u \) minus the number of 'holes'. As \( u \) increases the holes disappear and \( \chi_u \) counts the number of local maxima. So when \( u \) becomes close to the maximum of the random field \( Z_{\text{max}} \) we have that

\[
P(\text{reject } H_0 | H_0 \text{ true}) = P(Z_{\text{max}}) = P(\chi_u > 0) \approx E(\chi_u)
\]

where \( H_0 \) is the null hypothesis that there is no significant positive activation/signal present in the field. Thus the Type I error of the test can be controlled through knowledge of the Expected Euler characteristic.

**Note**: This function is directly copied from "AnalyzeFMRI".

**Value**

The value of the expected Euler Characteristic.
Example Datasets

Author(s)
J. L. Marchini

References

See Also
Threshold.RF

Examples
EC.3D(4.6, sigma = diag(1, 3), voxdim = c(1, 1, 1), num.vox = 10000)
EC.3D(4.6, sigma = diag(1, 3), voxdim = c(1, 1, 1), num.vox = 10000, type = "t", df = 100)

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.
toy1 and toy2 are two 3D toy examples.

Author(s)
Wei-Chen Chen and Ranjan Maitra.

References
False Discovery Rates for Spatial Signals

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**

```r
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**


**See Also**

`qvalue()`.
Examples

```r
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)
```

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

```r
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 similarity 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 dimensions `t * 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.
Generalized Cluster-Based Analysis (CBA) Method

Author(s)
Wei-Chen Chen.

References

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)

.rem <- function(){

### 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


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)

.rem <- function(){

### 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)
}

Likelihood Mixture Tests with Identity Cov Matrix or Only p-values

Likelihood Mixture Tests with Identity Cov Matrix or Only p-values
**Description**

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

**Usage**

```r
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 $H_0$: $fcobj.0$ vs. $Ha: fcobj.a$.

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

**Value**

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

**Author(s)**

Wei-Chen Chen and Ranjan Maitra.

**References**


See Also

EMCluster::lmt().

Examples

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

.rem <- function(){

### 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.

Procedure to adjust/select plausible p-values, proc = "1" uses q-value qvalue(), proc = "2" uses fdr.bh.p2(), and proc = "weight" uses a weighted version of fdr.bh.p2().

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.
Main functions

References


See Also

PARAM.

Examples

```r
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)

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

---

Main functions

Description

Main MixfMRI functions.

Usage

```r
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],
```
Arguments

- **X.gbd**: a data matrix of N voxel locations. \( \text{dim}(X.gbd) = N \times 3 \) for 3D data and \( N \times 2 \) for 2D data.
- **PV.gbd**: a p-value vector of signals associated with voxels. \( \text{length}(PV.gbd) = N \).
- **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.
Main functions

Beta density: The 1st component is restricted by \texttt{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: \texttt{model.X = "I"} is for diagonal cov matrix of multivariate Normal distribution, and "V" for unstructured cov matrix. \texttt{ignore.X = TRUE} is to ignore \texttt{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 \texttt{set.global()} has purposes: create a template/storage of parameters, save configurations, and called by \texttt{fclust()} to initial the parameters, such as \texttt{initial.em.gbd()} or \texttt{initial.RndEM.gbd()}.

Value

A list with class \texttt{fclust} by \texttt{fclust()} is returned which can be summarized by \texttt{print.fclust()}. A list \texttt{PARAM} or \texttt{PARAM.org} is returned by \texttt{set.global()}:

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

Author(s)

Wei-Chen Chen and Ranjan Maitra.
MixfMRI Control

Sets of controls in MixfMRI

Description

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

References


See Also

print.fclust().
MixfMRI Control

28

Format

Objects contain several parameters for methods.

Details

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

<table>
<thead>
<tr>
<th>Elements</th>
<th>Default</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>algorithm</td>
<td>&quot;apecma&quot;</td>
<td>implemented algorithm</td>
</tr>
<tr>
<td>optim.method</td>
<td>&quot;BFGS&quot;</td>
<td>optimization method</td>
</tr>
<tr>
<td>model.X</td>
<td>&quot;I&quot;</td>
<td>cov matrix structure</td>
</tr>
<tr>
<td>ignore.X</td>
<td>FALSE</td>
<td>if using voxel information</td>
</tr>
<tr>
<td>check.X.unit</td>
<td>TRUE</td>
<td>if checking X in [0, 1]</td>
</tr>
<tr>
<td>CONTROL</td>
<td>a list</td>
<td>see CONTROL next for details</td>
</tr>
<tr>
<td>INIT</td>
<td>a list</td>
<td>see INIT next for details</td>
</tr>
<tr>
<td>LRT</td>
<td>a list</td>
<td>see LRT next for details</td>
</tr>
<tr>
<td>MPI.gbd</td>
<td>FALSE</td>
<td>if MPI speedup available</td>
</tr>
<tr>
<td>common.gbd</td>
<td>TRUE</td>
<td>if X in common gbd format</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Elements</th>
<th>Default</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>min.1st.prop</td>
<td>0.8</td>
<td>minimum proportion of 1st cluster</td>
</tr>
<tr>
<td>max.PV</td>
<td>0.1</td>
<td>maximum p-value for initialization</td>
</tr>
<tr>
<td>BETA.alpha.min</td>
<td>0 + 1e-6</td>
<td>minimum value of alpha parameter of Beta distribution</td>
</tr>
<tr>
<td>BETA.alpha.max</td>
<td>1 - 1e-6</td>
<td>maximum value of alpha parameter of Beta distribution</td>
</tr>
<tr>
<td>BETA.beta.min</td>
<td>1 + 1e-6</td>
<td>minimum value of beta parameter of Beta distribution</td>
</tr>
<tr>
<td>BETA.beta.max</td>
<td>1e+6</td>
<td>maximum value of beta parameter of Beta distribution</td>
</tr>
<tr>
<td>max.try.iter</td>
<td>10</td>
<td>maximum retry iterations if result is unstable</td>
</tr>
<tr>
<td>class.method</td>
<td>&quot;prob.extned&quot;</td>
<td>classification method at initializations</td>
</tr>
</tbody>
</table>
The elements of LRT are default values or limitations for likelihood ratio tests including

<table>
<thead>
<tr>
<th>Elements</th>
<th>Default</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>H0.alpha</td>
<td>1</td>
<td>null hypothesis alpha parameter of Beta distribution</td>
</tr>
<tr>
<td>H0.beta</td>
<td>1</td>
<td>null hypothesis beta parameter of Beta distribution</td>
</tr>
<tr>
<td>H0.mean</td>
<td>0.05</td>
<td>null hypothesis mean of Beta distribution</td>
</tr>
</tbody>
</table>

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References


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 fclus().
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


See Also

set.global().

Examples

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

.rem <- function(){
    ### 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

Description
Print fclust 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().
Simulations

Value

A summary of fclust object is printed.

Author(s)

Wei-Chen Chen and Ranjan Maitra.

References


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, smooth = FALSE)

Arguments

phantom a phantom dataset.
overlap a desired overlap level.
smooth if gcv.smooth2d() be applied to the data.
Simulations

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


See Also

set.global().

Examples

```R
.reml <- function(){
  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


Summarized Overlap

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

Usage

summarized.overlap(overlap.mat)
**Threshold.Bonferroni**

_Calculates Bonferroni Threshold_

**Description**

Calculate the Bonferroni threshold for n iid tests that results in an overall p-value of p.val. The tests can be distributed as Normal, t or F.

**Usage**

Threshold.Bonferroni(p.val, n, type = c("Normal", "t", "F"), df1 = NULL, df2 = NULL)
**Threshold.FDR**

**False Discovery Rate (FDR) Threshold**

**Description**

Calculates the False Discovery Rate (FDR) threshold for a given vector of statistic values.

**Usage**

\[ \text{Threshold.FDR}(x, q, \text{cv.type} = 2, \text{type} = \text{c("Normal", "t", "F")}, \text{df1} = \text{NULL}, \text{df2} = \text{NULL}) \]
Arguments

- **x**: A vector of test statistic values.
- **q**: The desired False Discovery Rate threshold.
- **cV.type**: A flag that specifies the assumptions about the joint distribution of p-values. Choose cV.type = 2 for fMRI data (see Genovese et al (2001))
- **type**: The distribution of the statistic values. Either "Normal", "t" or "F".
- **df1**: The degrees of freedom of the t-distribution or the first degrees of freedom parameter for the F distribution.
- **df2**: The second degrees of freedom parameter for the F distribution.

Details

Note: This function is directly copied from "AnalyzeFMRI".

Value

Returns the FDR threshold.

Author(s)

J. L. Marchini

References

Genovese et al. (2001) Thresholding of Statistical Maps in Functional NeuroImaging Using the False Discovery Rate.

Examples

```r
x <- c(rnorm(1000), rnorm(100, mean = 3))
Threshold.FDR(x = x, q = 0.20, cV.type = 2)
```

Threshold.RF

Random Field Theory Thersholds.

Description

Calculates the Random Field theory threshold to give that results in a specified p-value.

Usage

```r
Threshold.RF(p.val, sigma, voxdim = c(1, 1, 1), num.vox,
        type = c("Normal", "t"), df = NULL)
```
**Arguments**

- `p.val` - The required p-value.
- `sigma` - The 3D covariance matrix of the random field.
- `voxdim` - The dimensions of a voxel.
- `num.vox` - The number of voxels that constitute the random field.
- `type` - The type of random field, "Normal" or "t".
- `df` - The degrees of the t distributed field.

**Details**

Calculates the threshold that produces an expected Euler characteristic equal to the required p-value.

Note: This function is directly copied from "AnalyzeFMRI".

**Value**

Returns the Random Field threshold.

**Author(s)**

J. L. Marchini

**See Also**

`EC.3D`

**Examples**

```r
a <- Threshold.RF(p.val = 0.05, sigma = diag(1, 3), voxdim = c(1, 1, 1), num.vox = 10000)
EC.3D(a, sigma = diag(1, 3), voxdim = c(1, 1, 1), num.vox = 10000)
```
Index

* datasets
  Example Datasets, 15
* global variables
  MixfMRI Control, 27
* package
  MixfMRI-package, 2
* programming
  algorithm, 3
  Compute Q values, 6
  Compute Statistics for Log Odds Ratio of Posterior Probability, 7
  Covariance Matrices, 9
  Covariance Matrices of Logit ETA, 11
  Density function of p-values, 13
  False Discovery Rates for Spatial Signals, 16
  Generalized Cluster-Based Analysis (CBA Method), 17
  initial, 19
  Likelihood Mixture Tests with Identity Cov Matrix or Only p-values, 20
  LRT, 22
  Main functions, 24
  Plotting, 29
  Print Objects, 31
  Simulations, 32
  Smoothing, 34
  Summarized Overlap, 34
* utilities
  cluster.threshold, 5
  EC.3D, 14
  Threshold.Bonferroni, 35
  Threshold.FDR, 36
  Threshold.RF, 37
  .FC.CT (MixfMRI Control), 27
  apecma.step.gbd (algorithm), 3
  cba.cor (Generalized Cluster-Based Analysis (CBA Method)), 17
  cluster.threshold, 5
  cluster_mass (cluster.threshold), 5
  Compute Q values, 6
  Compute Statistics for Log Odds Ratio of Posterior Probability, 7
  cov.logit.ETA (Covariance Matrices of Logit ETA), 11
  cov.logit.z (Covariance Matrices), 9
  cov.param (Covariance Matrices), 9
  cov.post.z (Covariance Matrices), 9
  Covariance Matrices, 9
  Covariance Matrices of Logit ETA, 11
  Density function of p-values, 13
  dmixpval, 7
dmixpval (Density function of p-values), 13
dpval, 7
dpval (Density function of p-values), 13
  EC.3D, 14, 38
  ecm.step.gbd (algorithm), 3
  em.step.gbd (algorithm), 3
  Example Datasets, 15
  False Discovery Rates for Spatial Signals, 16
  fclust, 3, 4, 20, 29–32
  fclust (Main functions), 24
  fdr.bh.p1, 18
  fdr.bh.p1 (False Discovery Rates for Spatial Signals), 16
  fdr.bh.p2, 18
  fdr.bh.p2 (False Discovery Rates for Spatial Signals), 16
  gcv.smooth2d (Smoothing), 34
gendataset, 13

gendataset (Simulations), 32

Generalized Cluster-Based Analysis (CBA) Method, 17

initial, 19

initial.em.gbd, 4, 26

initial.RndEM.gbd, 4, 26

Likelihood Mixture Tests with Identity Cov Matrix or Only p-values, 20

lmt.I (Likelihood Mixture Tests with Identity Cov Matrix or Only p-values), 20

lmt.pv (Likelihood Mixture Tests with Identity Cov Matrix or Only p-values), 20

logor.stat (Compute Statistics for Log Odds Ratio of Posterior Probability), 7

LRT, 22

lrt (LRT), 22

lrt2 (LRT), 22

Main functions, 24

MixfMRI (MixfMRI-package), 2

MixfMRI Control, 27

MixfMRI-package, 2

PARAM, 4, 20, 24

PARAM (Main functions), 24

PARAM.org, 4

plotfclust (Plotting), 29

plotfclustpv (Plotting), 29

plotpv (Plotting), 29

plotpvlegend (Plotting), 29

Plotting, 29

post.prob (Covariance Matrices), 9

Print Objects, 31

print.fclust, 26, 27

print.fclust (Print Objects), 31

pstats (Example Datasets), 15

pval.2d.complex (Example Datasets), 15

pval.2d.mag (Example Datasets), 15

qvalue, 13, 16

qvalue (Compute Q values), 6

set.global, 3, 4, 19, 20, 29, 30, 32, 33

set.global (Main functions), 24

shepp0fMRI (Example Datasets), 15

shepp1fMRI (Example Datasets), 15

shepp2fMRI (Example Datasets), 15

sheppAnat (Example Datasets), 15

Simulations, 32

Smoothing, 34

Summarized Overlap, 34

summarized.overlap (Summarized Overlap), 34

Threshold.Bonferroni, 35

Threshold.FDR, 36

Threshold.RF, 15, 37

toy1 (Example Datasets), 15

toy2 (Example Datasets), 15