Package: mritc (via r-universe)

October 19, 2026

Title MRI Tissue Classification

Date 2026-08-19

Version 0.6.2

Description Implements various methods for tissue classification in magnetic resonance (MR) images of the brain, including normal mixture models and hidden Markov normal mixture models, as outlined in Feng & Tierney (2011) <doi:10.18637/jss.v044.i07>. These methods allow a structural MR image to be classified into gray matter, white matter and cerebrospinal fluid tissue types.

Imports methods, oro.nifti

Suggests lattice, RNifti, misc3d, tinytest, covr

License GPL

URL https://github.com/jonclayden/mritc

BugReports https://github.com/jonclayden/mritc/issues

NeedsCompilation yes

Author Dai Feng [aut], Luke Tierney [aut], Jon Clayden [cre, aut] (ORCID: <https://orcid.org/0000-0002-6608-0619>)

Maintainer Jon Clayden <code@clayden.org>

Repository https://cran.r-universe.dev

Date/Publication 2026-08-20 05:50:02 UTC

RemoteUrl https://github.com/cran/mritc

RemoteRef HEAD

RemoteSha 49c14eb99cbde63d29a72a1fe5052df88194510c

Contents

mritc-package 2
emnormmix 4
initNormMix 5
makeMRIspatial 6

measureMRI 8
mritc 9
plot.mritc 13
print.mritc 14
readMRI 15
rnormmix 16
summary.mritc 17
writeMRI 17

Index **19**

`mritc-package` *MRI Tissue Classification Package*

Description

Use various methods to do MRI tissue classification.

Introduction

This package provides tools for MRI tissue classification using normal mixture models and hidden Markov normal mixture models (with the partial volume effect and intensity non-uniformity addressed) fitted by various methods.

Magnetic resonance imaging (MRI) is used to identify the major tissues within a subject's brain. Classification is usually based on a single image providing one measurement for each volume element, or voxel, in a discretization of the brain. A simple model for MRI tissue classification views each voxel as homogeneous, belonging entirely to one of the three major tissue types (cerebrospinal fluid (CSF), gray matter (GM), and white matter (WM)); the intensity of voxels are thus normally distributed with means and variances depending on the tissue types of their voxels. The tissue types are not known and need to be identified from the image. The assumption that all tissue types are independent leads to a simple normal mixture model with parameters estimated by the EM algorithm and tissue types assigned using the Bayes classifier.

Since nearby voxels tend to be of the same tissue type, a Markov random field model (a model from the Potts model family is used in this case) can be used to capture the spatial similarity of voxels by assigning homogeneity relationship among tissue types of neighboring voxels. Again, given the tissue types, the intensity of voxels are independently and normally distributed with means and variances depending on their tissue types. Furthermore, the Markov random field model defined on finite space is referred to as the hidden Markov model. Therefore the model combine the normal mixture part and the Potts model part is called the hidden Markov normal mixture model. This model can be fitted by the Iterated Conditional Mode algorithm, the Hidden Markov Random Field EM algorithm, or a Markov chain Monte Carlo approach.

A more realistic model than the one just described would take into account the fact that the volume elements are not homogeneous; while some may contain only one tissue type, others on the interface will contain two or possibly three different tissue types. This phenomenon is called the partial volume (PV) effect. One approach to address the PV effect is to introduce intermediate classes. Usually this is done by introducing two more classes: the combination of the CSF and the GM and the combination of the GM and the WM. Voxels containing WM and CSF are very rare and

are ignored. This helps reduce confounding in estimation and a number of studies have used this approach. Among these methods, the Gaussian partial volume hidden Markov random field models fitted by the modified EM algorithm appears to be more competitive in performance. A new approach to this problem is to construct a higher resolution image in which each voxel is divided into eight subvoxels. For each voxel the measured value is the sum of the unobserved measurements for the subvoxels. The subvoxels are in turn assumed to be homogeneous and follow the simpler model described above.

Intensity non-uniformity is an artifact that the signal intensity varies smoothly across an image. It is caused by combination and interaction of effects from the device, pulse sequence, and object. A commonly used approach to tackle it is to assume the the measured signal is equal to true signal multiplied by bias field associated with the intensity non-uniformity plus some noise. The bias field needs to be spatially smoothly varying and is modeled as either jointly normally distributed, or a linear combination of smooth spline or polynomial basis functions. Instead, we propose using a locally smoothed prior on the bias field.

A Bayesian hierarchical model aiming at modeling the partial volume effect and intensity non-uniformity simultaneously was proposed. Instead of splitting the task into different steps, the framework harmoniously integrates several sub-models addressing different issues in the MRI classification, through specification of the likelihood function and prior distributions. This approach could provide more accurate tissue classification and also allow more effective estimation of the proportion of each voxel that belongs to each of the major tissue types.

Besides brain image segmentation, the methods provided in this package can be used for classification of other spatial data as well.

Usage

The function `readMRI` and `writeMRI` are I/O functions for MRI data. Right now, the "Analyze", "NIfTI", and raw byte (unsigned with 1 byte per element in the byte stream) gzip formats are supported.

For each MR image, there has to be a corresponding array, `mask`, with values 1 and 0. Voxels with value 1 are inside the brain and 0 are outside. Tissue classification is conducted on voxels inside the brain.

The functions `mritc.em`, `mritc.icm`, `mritc.hmrfem`, and `mritc.bayes` are used to conduct the MRI tissue classification using the normal mixture model fitted by the EM algorithm, the hidden Markov normal mixture model at the voxel level fitted by the Iterated Conditional Mode algorithm, the Hidden Markov Random Field EM algorithm, or the Bayesian method (with or without the PV or bias field correction). The function `mritc.pvhmrfem` is for classification using Gaussian partial volume hidden Markov random field models fitted by the modified EM algorithm. Different components of the normal mixture model correspond to different tissue types. The number of components is flexible, say using five components model to address the PV effect by `mritc.em`, `mritc.icm`, `mritc.hmrfem`, or `mritc.bayes`.

In order to use the previous functions, the parameters of the normal mixture model and the Potts model have to be specified. Some parameters can be obtained using the functions `initOtsu` and `makeMRIspatial`. There are default values for other parameters.

The function `mritc` integrates all methods together, provides a uniform platform with easier usage, and generates an object of class "mritc" , for which generic functions `print.mritc`, `summary.mritc`, and `plot.mritc` are provided.

Computation Issues

To improve the speed, the table lookup method was used in various places; vectorized computation was used to take advantage of conditional independence. Some computations are performed by **C** code, and the **OpenMP** is used to parallelize key loops in the **C** code. Sparse matrix multiplication is adopted as well.

`emnormmix` *Estimate the Parameters of a Normal Mixture Model Using the EM Algorithm*

Description

Use the EM Algorithm to estimate the parameters of a normal mixture model.

Usage

```
emnormmix(y, prop, mu, sigma, err, maxit, verbose)
```

Arguments

`y`	vector of observations.
`prop`	vector of initial estimate of the proportions of different components of a normal mixture model.
`mu`	vector of initial estimate of the means of different components of a normal mixture model.
`sigma`	vector of initial estimate of the standard deviations of different components of a normal mixture model.
`err`	relative maximum error(s) used to decide when to stop the iteration. It could be a vector of length three corresponding to the relative maximum errors of the means, standard deviations, and proportions of all components of a normal mixture model. When it is a scalar, all have the same relative maximum error.
`maxit`	maximum number of iterations to perform.
`verbose`	logical. If `TRUE`, then indicate the level of output as the algorithm runs after every 10 iterations.

Details

It is tailor-made for the case when observations are from a finite set (MRI data for example). The table lookup method is used to speed up the computation.

Value

`prop`	a vector of estimated proportions of different components of a normal mixture model.
`mu`	a vector of estimated means of different components of a normal mixture model.
`sigma`	a vector of estimated standard deviations of different components of a normal mixture model.

Examples

```
prop <- c(0.3, 0.3, 0.4)
mu <- c(-10, 0, 10)
sigma <- rep(1, 3)
y<- floor(rnormmix(n=100000, prop, mu, sigma)[,1])
initial <- initOtsu(y, 2)
emnormmix(y=y, prop=initial$prop, mu=initial$mu, sigma=initial$sigma,
          err=1e-7, maxit=100, verbose=TRUE)
```

`initNormMix` *Get the Initial Estimate of the Parameters of a Normal Mixture Model*

Description

Obtain initial estimation of proportions, means, and standard deviations of different components (tissue types for MRI) based on threshold values generated by Otsu's method implemented by a fast algorithm, or proportion of different components.

Usage

```
initOtsu(y, m)
initProp(y, prop)
```

Arguments

`y`	a vector of intensity values of an image.
`m`	number of classes (tissue types for MRI) minus 1.
`prop`	the initial estimate of proportion of different components.

Details

The exhaustive search part of the function for Otsu's algorithm is adapted from `combn`. For `initProp`, the threshold values are quantiles based on the initial estimate of proportion of different components.

Value

`prop`	a vector of initial estimate of the proportions of different components (tissue types for MRI).
`mu`	a vector of initial estimate of the means of different components (tissue types for MRI).
`sigma`	a vector of initial estimates of the standard deviations of different components (tissue types for MRI).

Note

For initOtsu, it supports any number of m. However, for MRI data, it can be slow if m is bigger than 3 even with the fast algorithm implemented, since the Otsu's algorithm uses an exhaustive search. But it should be fine with m equal to 2, which corresponds to the typical case in MRI classification with three major tissue types CSF, GM, and WM.

References

Nobuyuki Otsu (1979). A threshold selection method from gray-level histograms *IEEE Transactions on Systems, Man and Cybernetics* **vol. 9**, 62-66

Ping-Sung Liao, Tse-Sheng Chen and Pau-Choo Chung (2001) A Fast Algorithm for Multilevel Thresholding *Journal of Information Science and Engineering* **vol. 17**, 713-727

Examples

```
#Example 1
prop <- c(.3, .4, .3)
mu <- c(40,  90, 130)
sigma <- c(4, 8, 4)
y <- floor(rnormmix(n=100000, prop, mu, sigma)[,1])
initOtsu(y, 2)
initProp(y, prop)

#Example 2
T1 <- readMRI(system.file("extdata/t1.rawb.gz", package="mritc"),
              c(91,109,91), format="rawb.gz")
mask <-readMRI(system.file("extdata/mask.rawb.gz", package="mritc"),
               c(91,109,91), format="rawb.gz")
initOtsu(T1[mask==1], 2)
initProp(T1[mask==1], c(0.17, 0.48, 0.35))
```

---

makeMRIspatial *Obtain Spatial Features of a Mask of an MR Image*

Description

Obtain various spatial features of an MR image, which are used in tissue classification.

Usage

```
makeMRIspatial(mask, nnei, sub, bias)
```

Arguments

mask	three dimensional array. The voxels with value 1 are inside the mask; with value 0 are outside. We just focus on voxels inside the mask.

nnei	the number of neighbors. Right now only 6, 18, and 26 neighbors are supported. For a 3D image, besides defining 6 neighbors in the x, y, and z directions, one can add 12 diagonal neighbors in the x-y, x-z, and y-z planes, and another 8 on the 3D diagonals. This leads to a six neighbor structure, an eighteen neighbor structure, and a twenty-six neighbor structure.
sub	logical; if TRUE, a new mask which splits each voxel into eight subvoxels is generated, and then obtain the neighbors and blocks of subvoxels; otherwise obtain the neighbors and blocks at the voxel level. The default if FALSE.
bias	logical; if TRUE, the spatial parameters for biased field correction are calculated. The default if FALSE.

Value

A list containing the following components:

neighbors	a matrix, each row of which giving the neighbors of a voxel or subvoxel. The number of rows is equal to the number of (sub)voxels within the mask and the number of columns is the number of neighbors of each (sub)voxel. For the (sub)voxels on the boundaries, when one or more of their neighbors are missing, the missing are represented by the total number of (sub)voxels within the mask plus 1.
blocks	the (sub)voxels within each block are mutually independent given the (sub)voxels in other blocks.
sub	logical; the same as the input sub.
subvox	if sub is TRUE, it is a matrix, with each row giving the eight subvoxels of a voxel; otherwise it is equal to NULL.
weights	if bias is TRUE, it is a vector of weights of neighbors of every voxel for bias field correction; otherwise it is equal to NULL. The default is NULL.
weineighbors	if bias is TRUE, it is a vector of sum of weights of neighbors for bias field correction, one element per voxel; otherwise it is equal to NULL. The default is NULL.

References

Dai Feng, Dong Liang, and Luke Tierney (2013) An unified Bayesian hierarchical model for MRI tissue classification *Statistics in Medicine*

Dai Feng (2008) Bayesian Hidden Markov Normal Mixture Models with Application to MRI Tissue Classification *Ph. D. Dissertation, The University of Iowa*

Examples

```
mask <- array(1, dim=c(2,2,2))
spa <- makeMRIspatial(mask, nnei=6, sub=FALSE)
spa <- makeMRIspatial(mask, nnei=6, sub=TRUE)
spa <- makeMRIspatial(mask, nnei=26, sub=TRUE, bias=TRUE)
```

`measureMRI` *Compare the Predicted Classsification Results with the Truth*

Description

Calculate and demonstrate different measures for classification results based on the truth.

Usage

```
measureMRI(intvec, actual, pre)
```

Arguments

`intvec`	a vector of intensity values. If it is not `NULL`, the density plots of each component corresponding to the actual and predicted classification results are shown. The default is `NULL`.
`actual`	matrix of the true classification result. Each row corresponds to one voxel. Column i represents the probabilities that all voxels are allocated to tissue type i.
`pre`	matrix of the predicted classification result. Each row corresponds to one voxel. Column i represents the probabilities that all voxels are allocated to tissue type i.

Value

`mse`	mean square error.
`misclass`	mis-classification rate.
`rseVolume`	root square error of volume with respect to reference tissue volume.
`DSM`	Dice Similary Measure of each tissue type. $$DSM^t_{a,b} = \frac{2 \times N^t_{a \cap b}}{N^t_a + N^t_b}$$ where N^t_a and N^t_b are the number of voxels classified as tissue type t by method a and b respectively, and $N^t_{a \cap b}$ is the number of voxels classified as tissue type t by both methods a and b. The larger the DSM, the more similar the results from the two methods.
`conTable`	confusion table. Each column of the table represents the instances in an actual class, while each row represents the instances in a predicted class.

Examples

```
#Example 1
prop <- c(.3, .4, .3)
mu <- c(-4, 0, 4)
sigma <- rep(1, 3)
y <- rnormmix(n=1e4, prop, mu, sigma)
```

```
intvec <- y[,1]
actual <- y[,2]
pre <- actual
pre[sample(1:1e4, 100, replace=FALSE)]  <- sample(1:3, 100, replace=TRUE)
actual <- do.call(cbind, lapply(1:3, function(i) ifelse(actual==i, 1, 0)))
pre <- do.call(cbind, lapply(1:3, function(i) ifelse(pre==i, 1, 0)))
measureMRI(intvec, actual, pre)


#Example 2
T1 <- readMRI(system.file("extdata/t1.rawb.gz", package="mritc"),
              c(91,109,91), format="rawb.gz")
mask <-readMRI(system.file("extdata/mask.rawb.gz", package="mritc"),
               c(91,109,91), format="rawb.gz")
tc.icm <- mritc(T1, mask, method="ICM")

csf <- readMRI(system.file("extdata/csf.rawb.gz", package="mritc"),
              c(91,109,91), format="rawb.gz")
gm <- readMRI(system.file("extdata/gm.rawb.gz", package="mritc"),
              c(91,109,91), format="rawb.gz")
wm <- readMRI(system.file("extdata/wm.rawb.gz", package="mritc"),
              c(91,109,91), format="rawb.gz")
truth <- cbind(csf[mask==1], gm[mask==1], wm[mask==1])
truth <- truth/255
measureMRI(T1[mask==1], truth, tc.icm$prob)
```

`mritc` *MRI Tissue Classification Using Various Methods*

Description

Conduct the MRI tissue classification using different methods including: the normal mixture model (NMM) fitted by the Expectation-Maximization (EM) algorithm; the hidden Markov normal mixture model (HMNMM) fitted by the Iterated Conditional Mode (ICM) algorithm, the Hidden Markov Random Field EM (HMRFEM) algorithm, or the Bayesian Markov chain Monte Carlo method (MCMC); the partial volume HMNMM fitted by the modified EM (PVHMRFEM) algorithm or the higher resolution HMNMM fitted by the MCMC method (MCMCsub); the HMNMM with both PV and intensity non-uniformity addressed (MCMCsubbias).

Usage

```
mritc.em(y, prop, mu, sigma, err, maxit, verbose)
mritc.icm(y, neighbors, blocks, spatialMat, beta, mu, sigma,
          err, maxit, verbose)
mritc.hmrfem(y, neighbors, blocks, spatialMat, beta, mu, sigma,
             err, maxit, verbose)
mritc.pvhmrfem(y, neighbors, blocks, spatialMat, beta, mu, sigma,
            err, maxit, verbose)
```

```
mritc.bayes(y, neighbors, blocks, sub, subvox,
       subbias, neighbors.bias, blocks.bias, weineighbors.bias, weights.bias,
            spatialMat, beta, mu, sigma, niter, verbose)
mritc(intarr, mask, method, verbose = TRUE)
```

Arguments

y	a vector of intensity values of voxels.
prop	a vector of initial estimate of the proportions of different components of a normal mixture model. It can be obtained using the function initOtsu.
mu	a vector of initial estimate of the means of different components of a normal mixture model. It can be obtained using the function initOtsu.
sigma	a vector of initial estimates of the standard deviations of different components of a normal mixture model. It can be obtained using the function initOtsu.
err	relative maximum error(s) used to decide when to stop the iteration. It could be a vector corresponding to the relative maximum errors of the means, standard deviations (for mritc.em, mritc.icm, mritc.hmrfem, and mritc.pvhmrfem), and proportions (for mritc.em) of all components of a normal mixture model. When it is a scalar, all have the same relative maximum error. The default value is 1e-4.
maxit	maximum number of iterations to perform. The default is 200 for mritc.em, 20 for mritc.icm, mritc.hmrfem, and mritc.pvhmrfem.
verbose	logical. If TRUE, then indicate the level of output as the algorithm runs.
neighbors	a matrix of neighbors of voxels. One row per voxel. It can be obtained using the function makeMRIspatial.
blocks	split voxels into different blocks to use the checker-board idea. It can be obtained using the function makeMRIspatial.
spatialMat	a matrix defining the spatial relationship in a Potts model. The default value is diag(1,3) for three components models for mritc.icm, mritc.hmrfem and mritc.bayes when sub is FALSE and matrix(c(2,0,-1,0,2,0,-1,0,2), nrow=3) when sub is TRUE. For mritc.pvhmrfem the default is matrix(c(2, 1, -1, -1, -1, 1, 2, 1, -1, -1, -1, 1, 2, 1, -1, -1, -1, 1, 2, 1, -1, -1, -1, 1, 2), ncol=5).
beta	the parameter 'inverse temperature' of the Potts model. The default value is 0.4 for mritc.icm, 0.5 for mritc.hmrfem, 0.6 for mritc.pvhmrfem. For mritc.bayes, the default is 0.7 when sub is FALSE and 0.3 when sub is TRUE.
sub	logical; if TRUE, use the higher resolution model; otherwise, use the whole voxel method.
subvox	for mritc.bayes, the match up tabel of voxels and their corresponding sub-voxels for the higher resolution model. It can be obtained using the function makeMRIspatial. For the whole voxel method, subvox=NULL
subbias	logical; if TRUE, use the model that addresses both the PV and intensity non-uniformity. The default is FALSE.
neighbors.bias	a matrix of neighbors of bias field. One row per voxel. It can be obtained using the function makeMRIspatial. The default is NULL.

`blocks.bias` blocks for bias field. It can be obtained using the function `makeMRIspatial`. The default is `NULL`.

`weineighbors.bias` a vector of sum of weights of neighbors of bias field. One element per voxel. It can be obtained using the function `makeMRIspatial`. The default is `NULL`.

`weights.bias` a vector of weights of different neighbors of every voxel. It can be obtained using the function `makeMRIspatial`. The default is `NULL`.

`niter` the number of iterations for `mritc.bayes`. The default values are 1000 and 100 for with and without bias field correction, respectively. The default values seem to be adequate in many cases.

`intarr` a three dimensional array of an MR image.

`mask` a mask of the MR image. Voxels with value 1 are inside the brain and value 0 are outside. Focus on voxels within the brain.

`method` a string giving the method for MRI tissue classification. It must be one of "EM", "ICM", "HMRFEM", "MCMC", "PVHMRFEM", "MCMCsub", or "MCMC-subbias" corresponding to using the NMM fitted by the EM algorithm; the HMNMM fitted by the ICM algorithm, the HMRFEM algorithm, or the MCMC; the partial volume HMNMM fitted by the PVHMRFEM algorithm; the higher resolution HMNMM fitted by the MCMC; the HMNMN addressing both the PV and intensity non-uniformity. It can be abbreviated. The default is "EM".

Details

The function `mritc` integrates functions `mritc.em`, `mritc.icm`, `mritc.hmrfem`, `mritc.pvhmrfem`, and `mritc.bayes`. It provides a uniform platform with easier usage. The user just need to specify the input MR image, the `mask` of the image, and the `method` used. The other parameters are specified automatically as follows. The parameters for the initial estimates of the proportions, means, and standard deviations of the normal mixture model are obtained using the function `initOtsu`. As to the parameters related to the Potts model, the six neighbor structure is used and then the `neighbors`, `blocks`, and `subvox` are obtained using the function `makeMRIspatial`. For the bias field correction, the twenty-six neighbor structure is used and then the `neighbors.bias`, `blocks.bias`, `weineighbors.bias` and `weights.bias` are obtained using the function `makeMRIspatial`. The other parameters are taken as the default values for each method. The process is reported during iterations.

Value

For `mritc`, it generates an object of class "mritc" which is a list containing the following components:

`prob` a matrix, one row per voxel and each column corresponding to the probabilities of being allocated to each component of a normal mixture model.

`mu` a vector of estimated means of the normal mixture model.

`sigma` a vector of estimated standard deviations of the normal mixture model.

`method` the method used for computation.

`mask` mask of an brain. Voxels inside it are classified.

Generic functions `print.mritc`, `summary.mritc`, and `plot.mritc` are provided.

For others, only `prob`, `mu`, and `sigma` are generated.

Note

The functions support the various normal mixture models with at most eight components.

References

Julian Besag (1986) On the statistical analysis of dirty pictures (with discussion) *Journal of the Royal Statistical Society. Series B (Methodological)* **vol. 48** 259-302

Meritxell Bach Cuadra, Leila Cammoun, Torsten Butz, Olivier Cuisenaire, and Jean-Philippe Thiran (2005) Comparison and validation of tissue modelization and statistical classification methods in T1-weighted MR brain images *IEEE Transactions on Medical Imaging*, **vol.24** 1548-1565

Dai Feng, Dong Liang, and Luke Tierney (2014) An unified Bayesian hierarchical model for MRI tissue classification *Statistics in Medicine* **vol.33, issue 8** 1349-1368

Dai Feng, Luke Tierney, and Vincent Magnotta (2012) MRI tissue classification using high resolution Bayesian hidden Markov normal mixture models *Journal of the American Statistical Association*, **vol.107, no.497** 102-119

Dai Feng and Luke Tierney (2011) mritc: A package for MRI tissue classification *Journal of Statistical Software*, **vol.44, no.7** 1-20 https://www.jstatsoft.org/v44/i07/

Dai Feng (2008) Bayesian hidden Markov normal mixture models with application to MRI tissue classification *Ph. D. Dissertation, The University of Iowa*

Yongyue Zhang, Michael Brady, and Stephen Smith (2001) Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm *IEEE Transactions on Medical Imaging* **vol. 20** 45-57

Examples

```
#Example 1
T1 <- readMRI(system.file("extdata/t1.rawb.gz", package="mritc"),
              c(91,109,91), format="rawb.gz")
mask <- readMRI(system.file("extdata/mask.rawb.gz", package="mritc"),
                c(91,109,91), format="rawb.gz")
y <- T1[mask==1]
initial <- initOtsu(y, 2)
prop <- initial$prop
mu <- initial$mu
sigma <- initial$sigma
tc.em <- mritc.em(y, prop, mu, sigma, verbose=TRUE)

## Not run:
mrispatial <- makeMRIspatial(mask, nnei=6, sub=FALSE)
tc.icm <- mritc.icm(y, mrispatial$neighbors, mrispatial$blocks,
                    mu=mu, sigma=sigma, verbose=TRUE)
tc.hmrfem <- mritc.hmrfem(y, mrispatial$neighbors, mrispatial$blocks,
                          mu=mu, sigma=sigma, verbose=TRUE)
tc.pvhmrfem <- mritc.pvhmrfem(y, mrispatial$neighbors, mrispatial$blocks,
                              mu=mu, sigma=sigma, verbose=TRUE)
```

```
  tc.mcmc <- mritc.bayes(y, mrispatial$neighbors, mrispatial$blocks,
                         mrispatial$sub, mrispatial$subvox,
                         mu=mu, sigma=sigma, verbose=TRUE)

  mrispatial <- makeMRIspatial(mask, nnei=6, sub=TRUE)
  tc.mcmcsub <- mritc.bayes(y, mrispatial$neighbors, mrispatial$blocks,
                         mrispatial$sub, mrispatial$subvox,
                         mu=mu, sigma=sigma, verbose=TRUE)

  mrispatial26 <- makeMRIspatial(mask, nnei=26, sub=TRUE, bias=TRUE)
  tc.mcmcsubbias <- mritc.bayes(y, mrispatial$neighbors, mrispatial$blocks,
                                mrispatial$sub, mrispatial$subvox,
                                subbias=TRUE, mrispatial26$neighbors,
                                mrispatial26$blocks,mrispatial26$weineighbors,
                                mrispatial26$weights, mu=mu, sigma=sigma, verbose=TRUE)


## End(Not run)
  #Example 2
  T1 <- readMRI(system.file("extdata/t1.rawb.gz", package="mritc"),
                c(91,109,91), format="rawb.gz")
  mask <-readMRI(system.file("extdata/mask.rawb.gz", package="mritc"),
                 c(91,109,91), format="rawb.gz")
  tc.icm <- mritc(T1, mask, method="ICM")
```

`plot.mritc` *Plot Method for Class "mritc"*

Description

Visualize MRI tissue classification results.

Usage

```
   ## S3 method for class 'mritc'
plot(x, method=c("RNifti","misc3d"), ...)
```

Arguments

`x`	an object of class "mritc"
`method`	the visualisation backend to use. The default, `"RNifti"`, uses `view`; the alternative, `"misc3d"`, uses the three-plane Tk-based viewer `slices3d`, which requires the **tkrplot tcltk** packages to be installed and useable.
`...`	any additional arguments for the relevant viewer.

Details

Allocate a voxel to the tissue type with the highest probability and then visualise the result.

Value

NULL, invisibly

Note

Prior to version 0.6.0, "misc3d" was the only viewer supported, but owing to its system requirements "RNifti" was added as a new default for better portability.

See Also

view, slices3d

Examples

```
T1 <- readMRI(system.file("extdata/t1.rawb.gz", package="mritc"),
              c(91,109,91), format="rawb.gz")
mask <-readMRI(system.file("extdata/mask.rawb.gz", package="mritc"),
               c(91,109,91), format="rawb.gz")
tc.icm <- mritc(T1, mask, method="ICM")
plot(tc.icm)
```

---

print.mritc *Print Method for Class "mritc"*

Description

Print out some information of an object of class "mritc".

Usage

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

Arguments

x	an object of class "mritc".
...	any additional arguments.

Value

The function computes and returns some summary statistics of the object obtained from running the function mritc.

Examples

```
T1 <- readMRI(system.file("extdata/t1.rawb.gz", package="mritc"),
              c(91,109,91), format="rawb.gz")
mask <-readMRI(system.file("extdata/mask.rawb.gz", package="mritc"),
               c(91,109,91), format="rawb.gz")
tc.icm <- mritc(T1, mask, method="ICM")
print(tc.icm)
```

`readMRI` *Read an MR Image into an Array*

Description

Read an MR image of different formats into an array.

Usage

```
readMRI(file, dim, format)
```

Arguments

`file`	the name of the image file to be read in.
`dim`	the dimensions of the image. It is required for the image of type `raw.gz`, in which `dim` is a vector of length three specifying dimensions in x, y, and z directions. The default is `NULL`.
`format`	the format of the image file. Right now only the "Analyze", "NIfTI", and raw byte (unsigned with 1 byte per element in the byte stream) gzip formats are supported.

Details

The files of "Analyze" format are read in through the function `readANALYZE`. The files of "NIfTI" format are read in through the function`readNIfTI`.

Value

An array with the appropriate dimensions containing the image volume.

See Also

`readANALYZE`, `readNIfTI`

Examples

```
## Not run:
vol1 <- readMRI("t1.rawb.gz", c(91,109,91), "rawb.gz")

vol2 <- readMRI("t1.nii.gz", format="nifti")

vol3  <- readMRI("t1.nii", format="nifti")

vol4 <- readMRI("t1-analyze.img", format="analyze")

## End(Not run)
```

---

rnormmix *Generate Random Samples from a Normal Mixture Model*

Description

Generate random samples from a normal mixture model.

Usage

```
rnormmix(n, prop, mu, sigma)
```

Arguments

n	number of observations.
prop	a vector of proportions of different components.
mu	a vector of means of different components.
sigma	a vector of standard deviations of different components

Value

A matrix with each row corresponding to one sample. The first column are sample values from a normal mixture model; the second column are the components from which observations come.

Examples

```
prop <- c(.17, .48, .35)
mu <- c(-4, 0, 4)
sigma <- rep(1, 3)
y <- rnormmix(n=10000, prop, mu, sigma)
hist(y[,1], breaks=50)
```

summary.mritc *Summary Method for Class "mritc"*

Description

Summarize some information of an object of class "mritc".

Usage

```
   ## S3 method for class 'mritc'
summary(object, ...)
```

Arguments

object	an object of class "mritc".
...	any additional arguments.

Value

The function computes and returns some summary statistics of the object obtained from running the function mritc.

Examples

```
T1 <- readMRI(system.file("extdata/t1.rawb.gz", package="mritc"),
              c(91,109,91), format="rawb.gz")
mask <-readMRI(system.file("extdata/mask.rawb.gz", package="mritc"),
               c(91,109,91), format="rawb.gz")
tc.icm <- mritc(T1, mask, method="ICM")
summary(tc.icm)
```

writeMRI *Write an MR Image*

Description

Write an MR image into a file of different formats.

Usage

```
writeMRI(data, file, header, format)
```

Arguments

data	MRI data in a three dimensional array or four dimensional array with the forth dimension equal to 1. It could be also an object of class "nifti" or "anlz" as defined in the package oro.nifti.
file	the name of the image file to be written out.
header	the header file. header is set as NULL for backward compatibility.
format	the format of the image file. Right now only the "Analyze", "NIfTI", and raw byte (unsigned with 1 byte per element in the byte stream) gzip formats are supported.

Details

Header file is not needed for the file of "Analyze" or "NIfTI" format anymore.

Files of "Analyze" format are written out through the function writeANALYZE. Files of "NIfTI" format are written out through the function writeNIfTI.

Value

Nothing is returned.

See Also

writeANALYZE, writeNIfTI

Examples

```
  ## Not run:
  writeMRI(vol, file="vol.rawb.gz", header=NULL, format="rawb.gz")

  writeMRI(vol, file="vol", header=NULL, format="nifti")

  writeMRI(vol, file="vol", header=NULL, format="analyze")

## End(Not run)
```

Index

∗ **classif**
  initNormMix, 5
  mritc, 9
∗ **distribution**
  rnormmix, 16
∗ **methods**
  plot.mritc, 13
  print.mritc, 14
  summary.mritc, 17
∗ **package**
  mritc-package, 2
∗ **spatial**
  makeMRIspatial, 6
∗ **utilities**
  emnormmix, 4
  measureMRI, 8
  readMRI, 15
  writeMRI, 17

combn, *5*

emnormmix, 4

initNormMix, 5
initOtsu, *3*, *6*, *10, 11*
initOtsu *(*initNormMix*)*, 5
initProp, *5*
initProp *(*initNormMix*)*, 5

makeMRIspatial, *3*, 6, *10, 11*
measureMRI, 8
mritc, *3*, 9, *11*, *14*, *17*
mritc-package, 2
mritc.bayes, *3*, *10, 11*
mritc.em, *3*, *10, 11*
mritc.hmrfem, *3*, *10, 11*
mritc.icm, *3*, *10, 11*
mritc.pvhmrfem, *3*, *10, 11*

plot.mritc, *3*, *12*, 13
print.mritc, *3*, *12*, 14

readANALYZE, *15*
readMRI, *3*, 15
readNIfTI, *15*
rnormmix, 16

slices3d, *13, 14*
summary.mritc, *3*, *12*, 17

view, *13, 14*

writeANALYZE, *18*
writeMRI, *3*, 17
writeNIfTI, *18*