Journal of Statistical Software

October 2011, Volume 44, Issue 7. *http://www.jstatsoft.org/*

mritc: A Package for MRI Tissue Classification

Dai Feng
Merck Research Laboratories

Luke Tierney
University of Iowa

Abstract

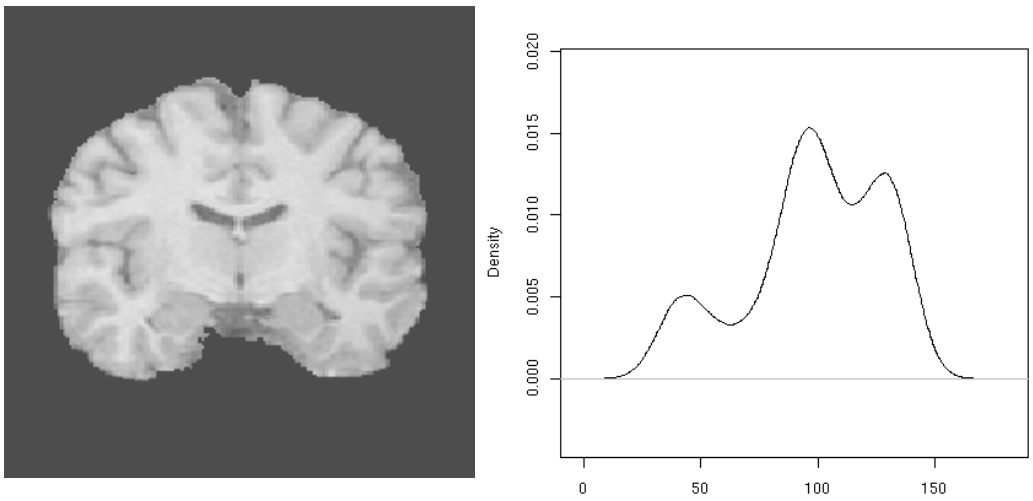
This paper presents an R package for magnetic resonance imaging (MRI) tissue classification. The methods include using normal mixture models, hidden Markov normal mixture models, and a higher resolution hidden Markov normal mixture model fitted by various optimization algorithms and by a Bayesian Markov chain Monte Carlo (MCMC) method. Functions to obtain initial values of parameters of normal mixture models and spatial parameters are provided. Supported input formats are ANALYZE, NIfTI, and a raw byte format. The function `slices3d` in **misc3d** is used for visualizing data and results. Various performance evaluation indices are provided to evaluate classification results. To improve performance, table lookup methods are used in several places, and vectorized computation taking advantage of conditional independence properties are used. Some computations are performed by C code, and OpenMP is used to parallelize key loops in the C code.

Keywords: higher resolution hidden Markov normal mixture model, Bayesian Markov chain Monte Carlo, table lookup, conditional independence, OpenMP.

1. Introduction

Magnetic resonance imaging (MRI) is a non-invasive method for imaging the inside of objects, for example different organs of animals and humans. Different types of images have different contrast effects to satisfy specific applications. Here, we focus on structural imaging of the brain. Each image is a three-dimensional (3D) array of image intensities, one for each voxel (volume picture element). In the package **mritc**, we focus on using signal intensities from a T1-weighted MR acquisition.

Figure 1(a) shows a coronal view of a brain. In a T1-weighted image, white matter (WM) tends to have light gray color, gray matter (GM) medium gray, and cerebrospinal fluid (CSF) dark gray (Hornak 2011). Brain tissue classification has many applications, for example in diagnosis of disease and preparation for surgery. Since manual tissue classification is labor intensive,



(a) Coronal view (b) Density of intensities

Figure 1: Coronal view of a brain from a T1-weighted image and its density plot of three tissues.

automated methods have been proposed to match the quality of the manual technique at lower cost.

MRI data consist of image intensities $y_1, \ldots, y_N$ for N voxels in a 3D array, where N is large, for example $256 \times 256 \times 192$. Typically, a binary mask is used to remove non-brain tissue, and the remaining number of voxels (within the brain mask) is still quite large. Intensities are often scaled to $[0, 255]$ and rounded to an integer. A density plot of intensities from a low noise MR image is shown in Figure 1 (b). For voxel $i = 1, \ldots, N$, the goal is to find the tissue types, denoted by $z_i \in \{1, 2, 3\}$ corresponding to the three major tissue types.

The motivation of the package was to make available some tissue classification methods especially the ones using the Bayesian MCMC based on Feng *et al.* (2012), which are not available elsewhere.

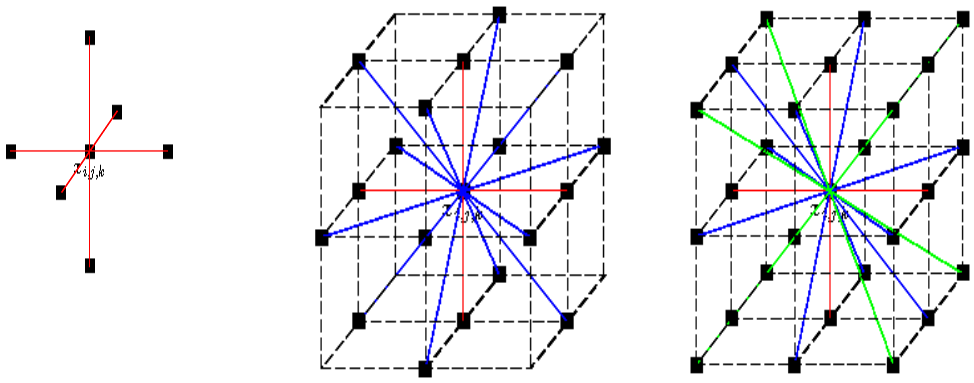
The next section reviews classification approaches provided in the package. Section 3 discusses computational issues involved to improve speed. The overview of the functions available and several examples are shown in Section 4. Some discussion and directions for future work are presented in Section 5.

2. Methods available

The package provides tools for MRI tissue classification using normal mixture models and (partial volume, higher resolution) hidden Markov normal mixture models fitted by various methods. This section outlines the approaches available. For more up-to-date references on MRI tissue classification, see Feng *et al.* (2012) and references therein.

2.1. The pure voxel assumption

There are three modes in Figure 1(b), and CSF are usually with small intensity values, the



(a) Six neighbors (b) 18 neighbors (c) 26 neighbors

Figure 2: Illustration of neighborhood structures in three dimensions.

GM are often in the middle, and WM are those that tend to have larger intensity values. A mixture model is a natural choice when there are multiple modes in a density plot. Given the tissue structure $\mathbf{z} = (z_1, \ldots, z_N)$, intensities are independent and normally distributed $y_i \,|\, z_i \sim N(\mu(z_i), \sigma^2(z_i))$. The means and variances depend on the tissue types. Assuming tissue types are independent leads to a simple normal mixture model (NMM)

$$f(\mathbf{y}) = \prod_{i=1}^{N} \sum_{z_i=1}^{k} \phi_{\mu(z_i),\sigma^2(z_i)}(y_i) p(z_i = k).$$

Parameters are easily estimated by the expectation-maximization (EM) algorithm (Dempster *et al.* 1977) and tissue types can be assigned using the Bayes classifier (Mitchell 1997).

Since the data come from a 3D array, a natural question would be how the spatial information could be incorporated. Since adjacent voxels are likely to contain the same tissue type, a more realistic model accounts for this spatial homogeneity in $\mathbf{z}$. The Potts model family provides simple models for spatial homogeneity

$$p(\mathbf{z}) = \frac{1}{C(\beta)} \exp\left\{ \sum_i \alpha_i(z_i) + \beta \sum_{i \sim j} w_{ij} f(z_i, z_j) \right\}, \tag{1}$$

where $C(\beta)$ is a normalizing constant and $i \sim j$ indicates neighboring voxels. We need to define neighborhood structure and then assign relationships among neighboring voxels. For a 3D array, besides defining six neighbors in the x, y, and z directions, one can add twelve diagonal neighbors in the $x - y$, $x - z$, and $y - z$ planes, and another eight on the 3D diagonals. This leads to a six-neighbor structure, an 18-neighbor structure, and a 26-neighbor structure (Figure 2).

The parameter β in (1), called inverse temperature, determines the level of spatial homogeneity between neighboring voxels in the image. A zero β would imply that neighboring voxels are independent. We use positive β values hereafter. The w_{ij} are weights and we assume $w_{ij} \equiv 1$ hereafter. The term $\sum_{i=1}^{N} \alpha_i(z_i)$ is called the *external field*. The $\alpha_i(z_i)$ are functions

of z_i. When $\beta = 0$, the external field completely characterizes the probability distribution of the independent $z_i,\ i = 1, 2, \ldots, N$.

The Potts model is an example of a Markov random field (MRF) model, since the conditional distribution of z_i given all others is equal to the conditional distribution of z_i given all its neighbors. Let I denote the indicator function and $\mathrm{I}(x, y) = 1$ when $x = y$ and 0 otherwise. When $f(z_i, z_j) = \mathrm{I}(z_i = z_j)$ model (1) becomes

$$p(\mathbf{z}) = \frac{1}{C(\beta)} \exp\left\{ \sum_{i=1}^{N} \alpha_i(z_i) + \beta \sum_{i \sim j} \mathrm{I}(z_i = z_j) \right\}. \tag{2}$$

For $k = 2$ components this model is called the Ising model (Ising 1925); for $k > 2$ it is the Potts (1953) model. The Ising model was originally proposed to describe the physical properties of magnets. Due to its power, the Ising model and its various versions have been widely used in other fields, such as brain models in cognitive science, information and machine learning theory (MacKay 2003), economics (Bourgine and Nadal 2004), sociology (Kohring 1996) and game theory (Hauert and Szabó 2005).

The most commonly used Potts model is the one without an external field and with $w_{ij} \equiv 1$,

$$p(\mathbf{z}) = \frac{1}{C(\beta)} \exp\left\{ \beta \sum_{i \sim j} \mathrm{I}(z_i = z_j) \right\}. \tag{3}$$

We refer to this as the *simple Potts model* hereafter.

After incorporating the spatial information, we have the hidden Markov normal mixture model (HMNMM)

$$p(\mathbf{y} \,|\, \mathbf{z}, \boldsymbol{\mu}, \boldsymbol{\sigma}^2)\, p(\mathbf{z}).$$

The spatial correlation is accommodated by $p(\mathbf{z})$, a model from the Potts family to characterize the property among tissue types of voxels, and given the tissue types, the intensity values are independent and normally distributed with means and variances depending on the tissue types.

The HMNMM can be fitted by the iterated conditional modes (ICM) algorithm which alternately maximizes each parameter conditional on all others being fixed (Besag 1986), or the hidden Markov random field EM (HMRFEM) algorithm which is a variation of the EM algorithm (Zhang *et al.* 2001). Though the ICM has its drawbacks, such as sensitivity to the initial configuration and visiting scheme, and getting stuck at a local maximum, it is fast and easy to implement. It is useful to run the ICM algorithm as a pilot study to make sure that the hidden Markov normal mixture model provides reasonable results before using more time consuming algorithms.

Alternatively, we can use the Bayesian MCMC method to fit the model by specifying a prior distributions $p(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ on $\boldsymbol{\mu}, \boldsymbol{\sigma}^2$ and then using MCMC to compute characteristics of the posterior distribution $p(\boldsymbol{\mu}, \boldsymbol{\sigma}^2, \mathbf{z} \,|\, \mathbf{y})$. Assume $\boldsymbol{\mu}, \boldsymbol{\sigma}^2, \mathbf{z}$ are independent; $\boldsymbol{\mu}$ have independent normal distributions with the ordering restriction $\mu_1 < \ldots < \mu_k$ to avoid identifiability issues associated with label switching (Celeux *et al.* 2000); and $\boldsymbol{\sigma}^2$ have independent and identically distributed inverse Gamma distributions. Then the full conditionals satisfy $\boldsymbol{\mu}$ independent normal, $\boldsymbol{\sigma}^2$ independent inverse Gamma, and $\mathbf{z}$ Potts model with external field

$\alpha_i(z_i) = \log f(y_i \,|\, \mu(z_i), \sigma(z_i))$. The MRI segmentation for human brain by the Bayesian MCMC method has rarely been used before. Almost all previously proposed Bayesian methods have used maximum a posteriori (MAP) estimation, which essentially reduces to an optimization problem (Feng *et al.* 2012).

2.2. Incorporating partial volume effects

In the previous section, all methods were based on the assumption that voxels are pure, containing only one tissue type. A more realistic model would take into account the fact that some voxels 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 (Pham *et al.* 2000).

To address this, one approach is to introduce intermediate classes: the combination of CSF/GM (CG) and the combination of GM/WM (GW). This helps reduce confounding in estimation. A number of studies have used this approach, and among them the Gaussian and partial volume (GPV) method, which uses normal mixture model with dependent means and variances, performs well (Cuadra *et al.* 2005). For the GPV, it assumes the means and variances of CG and GW are equal to the weighted average of corresponding pure tissues and the densities of voxels from CG and GW are equal to mean densities based on the distribution of weights. However, there is no analytical solution to the integrals and numerical integration must be used. The algorithm to implement that with a Potts prior is referred to as the partial volume HMRFEM (PVHMRFEM) in the package.

We have adopted a different approach. Each voxel is divided in half in the x, y, z directions, producing eight subvoxels. Each subvoxel is viewed as containing only one tissue type and the observed voxel intensity is equal to the sum of its eight unobserved. Conditional on the tissue types of subvoxels, the intensity values of subvoxels are independent normals and a spatial model is used at the subvoxel level. Furthermore, to capture the fact that CSF and WM rarely coexist in a voxel we use

$$p(\mathbf{z}) = \frac{1}{C(\beta_1, \beta_2)} \exp\left\{ \sum_{i \sim j} f(z_i, z_j) \right\},$$

where

$$f(z_i, z_j) = \begin{cases} \beta_1 & \text{if } z_i = z_j; \\ -\beta_2 & \text{if } \{z_i, z_j\} = \{\text{CSF, WM}\}; \\ 0 & \text{otherwise.} \end{cases}$$

We call this model the *repulsion Potts model* and basically it assumes that it is most likely that the neighboring subvoxels are of the same tissue type and it is most unlikely that one subvoxel is from CSF but its neighbor is from WM and vice versa. We use a Bayesian formulation to solve it. The higher resolution model fitted by the Bayesian MCMC (MCMCsub) provides more accurate tissue classification and also allows more effective estimation of the proportion of each voxel that belongs to each of the major tissue types. A similar subvoxel idea has been used in Van Leemput *et al.* (2003). However, there are differences between their approach and ours; in particular, our method avoids problems such as classifying voxels containing both WM and GM as pure GM voxels (Feng *et al.* 2012).

3. Computational issues

The size of the MRI data is typically very large, for example, the data from BrainWeb (Collins *et al.* 1998), a simulated brain database, has over 1.8 million voxels even after masking out non-brain voxels. Furthermore when we use the higher resolution model in Section 2.2, each voxel is further divided into eight subvoxels. To improve speed, table lookup methods are used in various places, vectorization is used to take advantage of conditional independence, and some computations are performed by C code and C using OpenMP to parallelize loops.

3.1. Table lookup

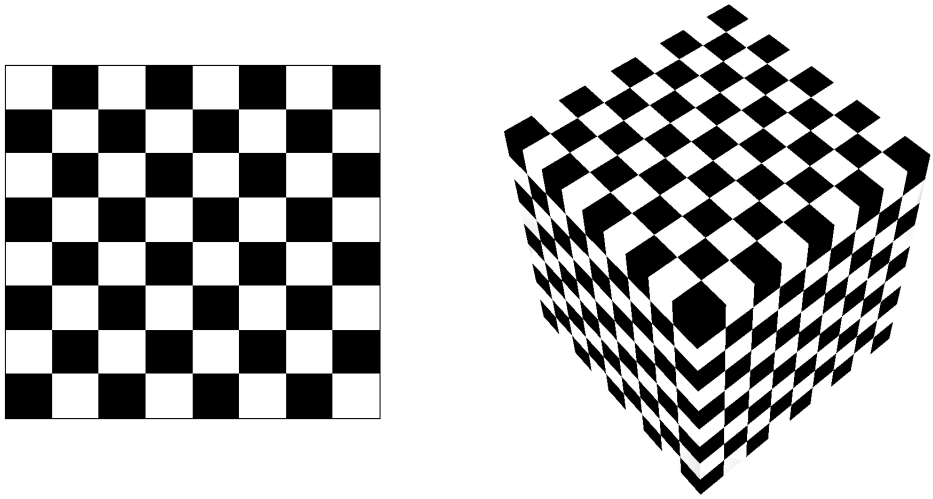
Table lookup methods use a table of pre-computed values for complex expressions and replace an expensive fresh computation of the value then it is needed by a fast indexing operation into the table of pre-computed values. This approach is useful when the number of distinct values is small and an appropriate table index is easily computed. Tang (1991) describes the use of table lookup methods for various elementary functions.

As mentioned in Section 1, the intensity values are integers between 0 and 255 and hence when we calculate the density values for a normal distribution at these intensities we only need a table with at most 256 values corresponding to intensity values $0, 1, 2, \ldots, 255$ to handle all voxels.

For a model from the Potts family, the conditional tissue type for a (sub)voxel is defined uniquely by its neighbors, and the conditional distributions are exactly the same discrete distribution given the same configuration of (sub)voxels in the neighboring voxels. Furthermore, the conditional distribution depends only on the total number of (sub)voxels from different tissue types. Therefore we can enumerate all possible cases and their corresponding conditional probabilities and use a pre-computed table of probabilities for all (sub)voxels when updating tissue types. For example when each (sub)voxel has six neighbors and there are three tissue types, there are only 343 different cases in the table. To enumerate all possible cases, first all configurations for different numbers of neighbors and tissue types are computed using a recursive method and then the unique combination of total number of (sub)voxels from different tissue types are generated. For the commonly used default setup with three tissue types and either six, 18, or 26 neighbors, the tables are pre-computed and saved for repeated usage. For others, the tables are computed at runtime.

3.2. Conditional independence

Suppose we can divide variables that need to be updated into disjoint blocks and, given the variables in other blocks, all the variables within the same block are conditionally independent. Then we can update one block at a time, and generate variables within each block independently. This may allow the generation within a block to be vectorized or parallelized. For example, in Figure 3(a), under the four neighbor structure in 2D (a voxel has neighbors on its south, north, east, and west), given the blacks, the whites are independent and vice versa. By this kind of independence, the vector $\mathbf{z}$ (indices to tissue types) can be updated in two steps: one for the blacks, one for the whites. This concept can be traced back to Besag's "Coding Methods" (Besag 1974); it was described in Wilkinson (2005) and a detailed discussion can be found in Winkler (2003). This conditional independence can be extended to 3D lattices with a six neighbor structure. In Figure 3(b), given the blacks, the whites are



(a) Four neighbors in 2D. (b) Six neighbors in 3D.

Figure 3: Illustration of conditional independence.

independent and vice versa. The minimum number of blocks to make the variables within each block independent given the other blocks is called the *chromatic number* (Winkler 2003). So the chromatic numbers for six neighbors in 3D is two, for 18 neighbors it is seven, and for 26 it is eight. See Feng (2008) for more details.

The conditional independence is used for vectorized computation when updating the vector $\mathbf{z}$. Combined with the table lookup method, the speed is greatly improved. To further improve the speed, embedded C code with parallel computation implemented by OpenMP is used.

3.3. OpenMP

OpenMP is a framework for shared memory parallel computing. It is supported by most C/C++ and Fortran compilers and runtime systems currently in use, and is based on the fork/join model. For a concise introduction to OpenMP see Tierney (2007).

The following code snippet illustrates the usage of the OpenMP in a relatively simple case. The code is from an embedded C function to fulfill the task of updating (sub)voxel intensity values. The upper part is an abstract of the function from the sequential code and the lower part illustrates its corresponding parallel code. We can see that the additional coding effort is not very much in this case.

```
/* an abstract of the function from the sequential code */
for (i = 0; i < n; i++) {
}

/* the abstract of corresponding papallel code */
#pragma omp parallel for firstprivate(k, ldD, ...)
for (i = 0; i < n; i++) {
}
```

Number of processors	1	2	3	4	5	6	7	8
Time	961	710	604	538	510	487	470	450

Table 1: The computational time in seconds of using the MCMCsub method on a full BrainWeb data set.

One of the advantages of OpenMP is that we can upgrade the code from sequential to parallel one step at a time. In each step, we just focus on part of the code. In this way, there is an incremental path to parallelism and the integrity of typically well tested serial code is maintained.

To realize parallelism, compiler directives `#pragma` in C are used and the keyword `omp` distinguishes the `pragma` as an OpenMP `pragma`. We parallelize a for-loop specified by `parallel for`. With a compiler that can recognize the directive, the code is run in a parallel fashion, otherwise the code is run sequentially. Therefore it is easy to port the code from one machine to another.

In the example, there is also a scope clause `firstprivate` to specify the sharing attributes of variables and an implicit barrier for synchronization. From our experience, it is a good idea to explicitly specify the scope of all variables used in a parallel region. In particular it is important to specify the `firstprivate` scope for variables that are initialized in the sequential portion and used but not modified in the parallel loop. Failure to identify these as `firstprivate` can result in poor performance. Some compilers may be able to infer this property in some cases but others are not able to, so to ensure that code works well on a range of compilers explicit specification is a good idea. See Chandra *et al.* (2001) for detailed discussion on OpenMP.

The computational time of using the MCMCsub method on a full BrainWeb data set (with 1,838,675 voxels inside the mask) is shown in Table 1. The number of sweeps is 100, which seems to be sufficient to produce good classifications. The speedup in shown in Figure 4. The program is run on a Linux dual quad-core AMD Opteron 2.3GHz, 48GB RAM computer. The speedup is about 2.1 when eight processors are used. By Amdahl's law, no matter how well the parallel proportion is coded and how many processors are used, the performance of the overall code will be limited by the proportion that can be parallelized and the larger the number of processors, the more severe the effect.

For the number of processors, without explicit specification, all available ones are used. To specify the number of processors, we can use the environment variable `OMP_NUM_THREADS`. Suppose the code is run with four processors. For the Unix-like machines, run R with `OMP_NUM_THREADS` specified as follows.

```
OMP_NUM_THREADS=4 R
```

For the Windows users, the `OMP_NUM_THREADS` can be specified as follows. First, click "start" to find R icon. Second, right click the icon and then click "properties". Finally, add `OMP_NUM_THREADS=4` to the "Target". Furthermore, for the Windows version, one needs to put `pthreadGC2.dll` at `C:\WINDOWS\system32`. To obtain `pthreadGC2.dll`, use `ftp://sourceware.org/pub/pthreads-win32/dll-latest/lib/pthreadGC2.dll`. Functions to test and specify the number of processors within the package may be provided in the future.

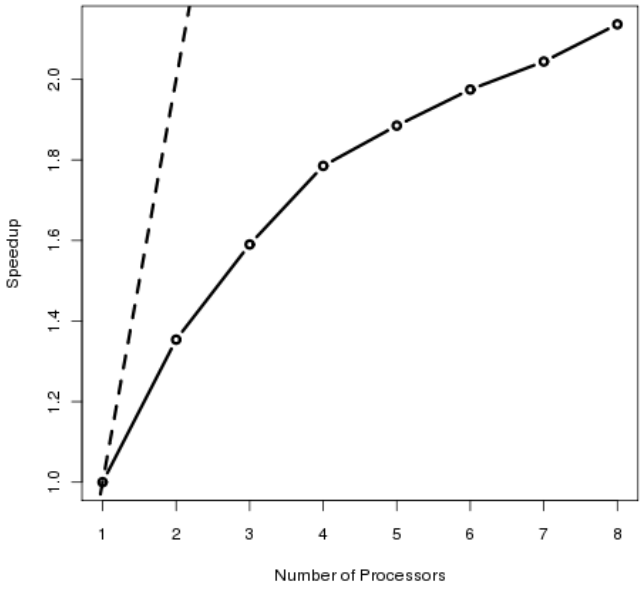


Figure 4: Speedup of the OpenMP example. The solid line shows the speedup and the dashed line is produced using `abline(0, 1)`.

4. Overview of the package

In this section, we give first an overview of the different functions of the package and then several examples on the usage of the package in detail.

4.1. Functions of the package

The ANALYZE, NIfTI, and raw byte file formats are supported for input and output. For the ANALYZE format, the functions are taken from the package **AnalyzeFMRI** (Marchini and Lafaye de Micheaux 2009; Bordier *et al.* 2011). For the NIfTI format, the functions are adapted from the package **fmri** (Tabelow and Polzehl 2011). For input, in addition to the original `.nii` files and the `.hdr`/`.img` files, the `.nii.gz` files are supported. For output instead of `.nii` the adapted version writes data into compressed `.nii.gz` files to save space.

Initial values of the means, variances, and proportions of the NMMs can be generated using Otsu's method implemented by a fast algorithm using the function `initOtsu`. Another function to generate initial estimates is `initProp`. Otsu's algorithm was originally proposed to threshold a gray-level image to a binary image (Otsu 1979). It was generalized to multi-level thresholding and a fast algorithm was proposed in Liao *et al.* (2001). Note that for MRI data, it can be slow if the number of classes is greater than three (more than two threshold values) even with the fast algorithm implemented, since Otsu's algorithm uses an exhaustive search. But it should be sufficient with three classes, which corresponds to the typical case in MRI classification. For `initProp`, the threshold values are quantiles based on the guess of proportion of different components which is much faster than using `initOtsu` when there are more than three components.

Function	Method
`mritc.em`	NMM fitted by the EM algorithm
`mritc.icm`	HMNMM fitted by the ICM algorithm
`mritc.hmrfem`	HMRFEM algorithm
`mritc.bayes` with `sub = FALSE`	MCMC
`mritc.pvhmrfem`	PVHMRFEM algorithm
`mritc.bayes` with `sub = TRUE`	MCMCsub

Table 2: The classification methods and corresponding function names.

Various spatial input parameters for different methods, such as the neighbors of corresponding voxels, the blocks for conditional independence, and the subvoxels corresponding to each voxel, can be obtained using the function `makeMRIspatial`. The package supports six, 18, and 26 neighbors structure typical for MRI (Figure 2).

The classification methods and their corresponding function names in the package are listed in Table 2.

For the default setup of the Potts models, a simple Potts model is used. The exception is for MCMCsub and PVHMRFEM, where the repulsion Potts model is adopted. A six neighbor structure in 3D is the default neighbor definition. The default value of β for PVHMRFEM is taken from Cuadra *et al.* (2005). For other methods, the values are obtained through a validation procedure (Feng *et al.* 2012). The argument `spatialMat` is used to specify the format of $f(z_i, z_j)$ in the model (1). For a simple Potts model, it is an identity matrix and for a repulsion Potts model it is used to specify β_1/β and $-\beta_2/\beta$.

The package provides a uniform platform for all methods with easier usage by the function `mritc`. The user must specify the input MR image `intarr`, the `mask` of the image, and the `method` used. The other parameters are defined automatically. The output is an object of class `mritc` and generic `print`, `summary`, and `plot` methods are provided.

Furthermore different metrics for accuracy of predictions based on available truth can be obtained through the function `measureMRI`. The measures available include: mis-classification rate, average mean square error, dice similarity measure (DSM, Cuadra *et al.* 2005), confusion table, and tissue volume error. The mis-classification rate compares whole voxel classification to the discrete anatomical model. This requires a form of rounding of the results and doesn't reflect all the PV information each approach provides. To demonstrate the ability of capturing the PV effect, average mean square errors between the estimated and the true tissue distributions within voxels is provided. Another performance measure is the DSM defined as follows

$$DSM^t_{a,b} = \frac{2N^t_{a\cap b}}{N^t_a + N^t_b},$$

where N^t_a and N^t_b are the number of voxels classified as tissue t by method a and b respectively, and $N^t_{a\cap b}$ is the number of voxels classified as tissue t by both methods a and b. The larger the DSM, the more similar the results from the two methods. Furthermore, the confusion table could be used to evaluate the results on a tissue per tissue base. The element on the ith row and jth column of the table is the probability of classifying voxels as being from category i given that the true category should be j. The differences in absolute values between the estimated and true tissue volumes with respect to the truth are also available.

4.2. Examples

This part demonstrates the usage of the package through four examples using the data sets contained in the package. The data was adapted from the BrainWeb repository with the size reduced to take less space and time for the purpose of illustration.

The first example reads the T1-weighted image and its corresponding mask using `readMRI` and focus on voxels inside the brain. The classification is based on the NMM fitted by the EM algorithm using `mritc.em`. The initial values of the NMM are obtained using `initOtsu`.

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

Iteration 10: relative error of mu = 8e-04; sigma = 0.006; prop = 0.002
Iteration 20: relative error of mu = 5e-04; sigma = 0.004; prop = 0.001
Iteration 30: relative error of mu = 4e-04; sigma = 0.003; prop = 9e-04
Iteration 40: relative error of mu = 3e-04; sigma = 0.002; prop = 7e-04
Iteration 50: relative error of mu = 2e-04; sigma = 0.001; prop = 5e-04
Iteration 60: relative error of mu = 1e-04; sigma = 0.001; prop = 4e-04
Iteration 70: relative error of mu = 1e-04; sigma = 8e-04; prop = 3e-04
Iteration 80: relative error of mu = 8e-05; sigma = 6e-04; prop = 2e-04
Iteration 90: relative error of mu = 6e-05; sigma = 5e-04; prop = 2e-04
Iteration 100: relative error of mu = 4e-05; sigma = 4e-04; prop = 1e-04
Iteration 110: relative error of mu = 3e-05; sigma = 3e-04; prop = 9e-05
Iteration 120: relative error of mu = 2e-05; sigma = 2e-04; prop = 7e-05
Iteration 130: relative error of mu = 2e-05; sigma = 2e-04; prop = 5e-05
Iteration 140: relative error of mu = 1e-05; sigma = 1e-04; prop = 4e-05

R> str(tc.em)

List of 3
 $ prob : num [1:229786, 1:3] 1 0.999 1 0.924 0.994 ...
 $ mu   : num [1:3] 45.1 96.9 131
 $ sigma: num [1, 1:3] 11.47 15 9.92
```

By specifying the option `verbose = TRUE`, the function `mritc.em` indicates the convergence level as it runs. The output is a list of three components. The components `mu` and `sigma` are the estimates of mean and standard deviation (sd) vectors of the normal mixture model. The component `prob` is a matrix with one row per voxel and the number of columns equal to the number of tissue types. The value at the ith row and $j$the column is the probability

that voxel i is from tissue type j times the density of voxel i from the jth component of the normal mixture model.

Instead of using the HMM, the second example use the higher resolution HMNMM fitted by the Bayesian method. The spatial parameters are obtained using the function `makeMRIspatial` with argument `sub = TRUE`.

```
R> T1 <- readMRI(system.file("extdata/t1.rawb.gz", package = "mritc"),
+    c(91, 109, 91), format = "rawb.gz")
R> mask <- readMRI(system.file("extdata/mask.rawb.gz", package = "mritc"),
+    c(91, 109, 91), format = "rawb.gz")
R> y <- T1[mask == 1]
R> initial <- initOtsu(y, 2)
R> mu <- initial$mu
R> sigma <- initial$sigma
R> mrispatial <- makeMRIspatial(mask, nnei = 6, sub = TRUE)
R> tc.mcmcsub <- mritc.bayes(y, mrispatial$neighbors, mrispatial$blocks,
+    mrispatial$sub, mrispatial$subvox, mu = mu, sigma = sigma,
+    verbose = TRUE)

Iteration 10 has finished
Iteration 20 has finished
Iteration 30 has finished
Iteration 40 has finished
Iteration 50 has finished
Iteration 60 has finished
Iteration 70 has finished
Iteration 80 has finished
Iteration 90 has finished
Iteration 100 has finished

R> str(tc.mcmcsub)

List of 3
 $ prob : num [1:229786, 1:3] 0.964 0.887 0.975 0.7 0.789 ...
 $ mu   : num [1:3] 4.07 11.92 17.01
 $ sigma: num [1:3] 2.57 2.26 2.68
```

By supplying the option `verbose = TRUE`, the function `mritc.bayes` indicates the progress. The output is a list of three components consisting of posterior estimate.

Compared with the first two examples, the third is more comprehensive. The T1-weighted image and the mask are read in and the image is plotted using the function `slices3d` in **misc3d** (Feng and Tierney 2008) and then the wrapper of all classification methods `mritc` is used to classify the image using the ICM algorithm. Next, the classification results are plotted and evaluated by different metrics based on the truth. The truth reflects the proportion of different tissues present in each voxel. Finally, the images of the classification results and the original are overlaid for an overall view.

First the image is read and plotted.

```
R> T1 <- readMRI(system.file("extdata/t1.rawb.gz", package = "mritc"),
+    c(91, 109, 91), format="rawb.gz")
R> mask <-readMRI(system.file("extdata/mask.rawb.gz", package="mritc"),
+    c(91, 109, 91), format = "rawb.gz")
R> slices3d(T1)
```

Second, the image is classified and the classification results are plotted. The function `mritc` by default outputs the progress.

```
R> tc.icm <- mritc(T1, mask, method = "ICM")

Iteration 1: relative error of mu = 0.003; sigma = 0.02
Iteration 2: relative error of mu = 0.001; sigma = 0.007
Iteration 3: relative error of mu = 4e-04; sigma = 0.002
Iteration 4: relative error of mu = 4e-05; sigma = 2e-04
Iteration 5: relative error of mu = 0; sigma = 0

R> plot(tc.icm)
```

Next the truth is read, transformed to proportions of different types per voxel, and then the function `measureMRI` is used to evaluate the classification results.

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

$mse
[1] 0.03023166
$misclass
[1] 0.0889828
$rseVolume
[1] 0.06984997 0.03455316 0.01421461
$DSM
[1] 0.9241746 0.9060223 0.9112816
$conTable
          [,1]       [,2]       [,3]
[1,] 0.9564514 0.03925961 0.00000000
[2,] 0.0435486 0.89036938 0.08224165
[3,] 0.0000000 0.07037101 0.91775835
```

The classification results are good based on different metrics of accuracy. For example the DSM is about 0.92, 0.91, and 0.91 for CSF, GM, and WM respectively.

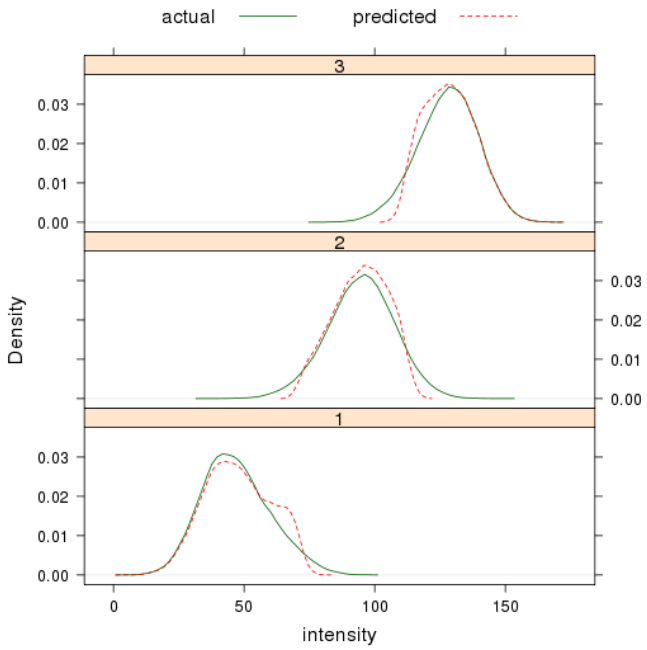


Figure 5: Density plots based on the true and estimated classification results.

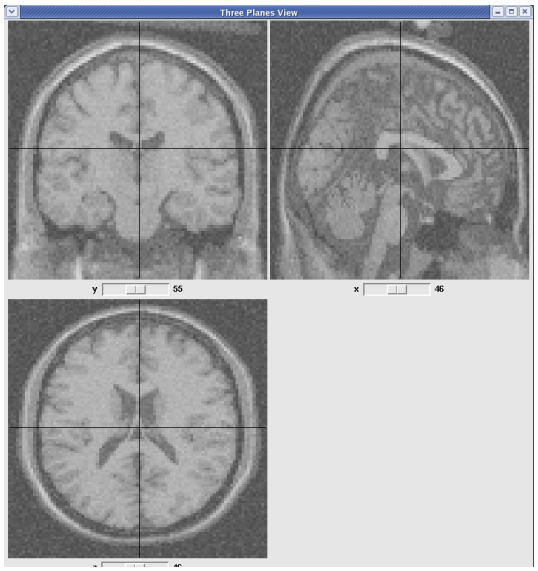
When the intensity values of voxels are supplied as input, the function `measureMRI` outputs the plots of density functions based on the true and estimated classification results. Figure 5 shows the output density plots.

Finally we overlay the images of the classification results and the original. The plots of the T1-weighted image, the classification results, and the overlay of the images are shown in Figure 6(a), (b), and (c) respectively. The WM is in white in Figure 6 (b) and dark green in Figure 6(c); GM is in light grey in Figure 6(b) and yellow green in Figure 6(c); the CSF is dark grey in Figure 6(b) and sandy brown in Figure 6(c). The voxels in black in Figure 6(b) and white in Figure 6(c) are outside of brain.

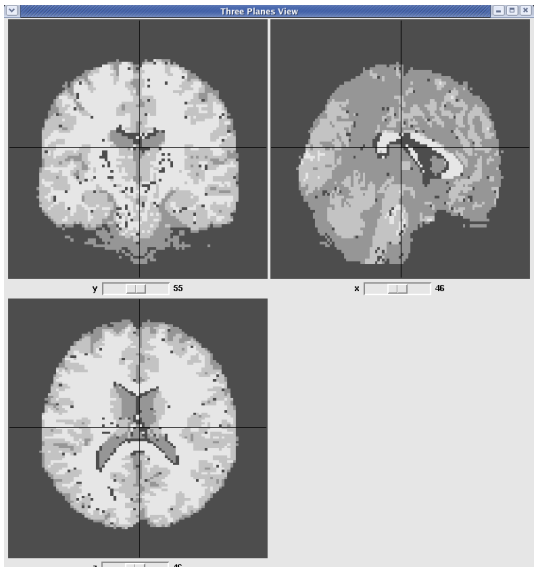
```
R> t1 <- T1
R> t1[mask == 1] <- 0
R> icm.class <- max.col(tc.icm$prob)
R> tc.icm$mask[tc.icm$mask == 1] <- icm.class
R> slices3d(t1, tc.icm$mask, col2 = terrain.colors(4)[4:1], alpha = 0.7)
```

In the last example, we show different ways to obtain initial estimates. As mentioned in Section 4.1, there are two functions available. Compared with `initOtsu`, which just needs the specification of number of threshold values, `initProp` requires a rough guess on proportions of different tissues. As to the speed, the former is slower than the later, though the difference is small when there are three types of tissues.

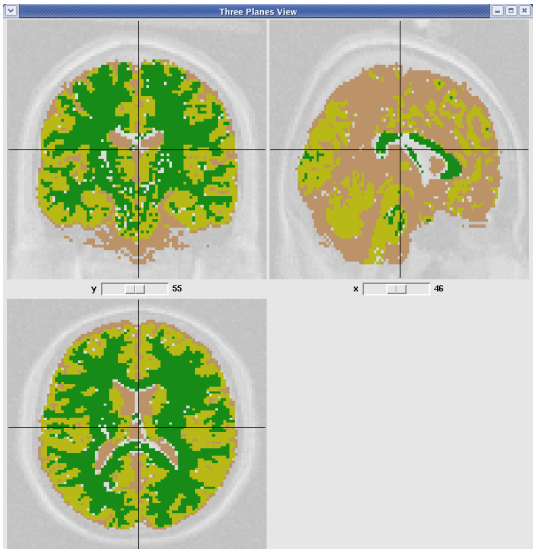
```
R> T1 <- readMRI(system.file("extdata/t1.rawb.gz", package = "mritc"),
+    c(91, 109, 91), format = "rawb.gz")
R> mask <- readMRI(system.file("extdata/mask.rawb.gz", package = "mritc"),
+    c(91, 109, 91), format = "rawb.gz")
```



(a) T1 image



(b) Classification results



(c) Overlay

Figure 6: The original T1-weighted image, its classification results by using the ICM algorithm to fit the HMNMM, and the overlay of the two.

```
R> y <- T1[mask == 1]
R> mrispatial <- makeMRIspatial(mask, nnei = 6, sub = TRUE)
R> system.time(initial.otsu <- initOtsu(y, 2))

   user  system elapsed
  2.205   0.069   2.263

R> mu.otsu <- initial.otsu$mu
R> sigma.otsu <- initial.otsu$sigma
```

```
R> tc.mcmcsub.otsu <- mritc.bayes(y, mrispatial$neighbors,
+    mrispatial$blocks, mrispatial$sub, mrispatial$subvox, mu = mu.otsu,
+    sigma = sigma.otsu, verbose = FALSE)
R> prop <- c(0.17, 0.48, 0.35)
R> system.time(initial.prop <- initProp(y, prop))

   user  system elapsed
  0.198   0.000   0.198

R> mu.prop <- initial.prop$mu
R> sigma.prop <- initial.prop$sigma
R> tc.mcmcsub.prop <- mritc.bayes(y, mrispatial$neighbors,
+    mrispatial$blocks, mrispatial$sub, mrispatial$subvox, mu = mu.prop,
+    sigma = sigma.prop, verbose = FALSE)
R> prop2 <- c(0.15, 0.52, 0.33)
R> initial.prop2 <- initProp(y, prop2)
R> mu.prop2 <- initial.prop2$mu
R> sigma.prop2 <- initial.prop2$sigma
R> tc.mcmcsub.prop2 <- mritc.bayes(y, mrispatial$neighbors,
+    mrispatial$blocks, mrispatial$sub, mrispatial$subvox, mu = mu.prop2,
+    sigma = sigma.prop2, verbose = FALSE)
R> measureMRI(y, truth, tc.mcmcsub.otsu$prob)

$mse
[1] 0.02562679
$misclass
[1] 0.08622806
$rseVolume
[1] 0.04657534 0.04331566 0.03727000
$DSM
[1] 0.9255861 0.9124067 0.9101404
$conTable
             [,1]       [,2]      [,3]
[1,] 9.040313e-01 0.01710043 0.0000000
[2,] 9.594260e-02 0.93216741 0.1068200
[3,] 2.609263e-05 0.05073217 0.8931800

R> measureMRI(y, truth, tc.mcmcsub.prop$prob)

$mse
[1] 0.02559359
$misclass
[1] 0.08665889
$rseVolume
[1] 0.04649706 0.05065990 0.04737376
$DSM
```

```
[1] 0.9256024 0.9122843 0.9090413
$conTable
             [,1]       [,2]      [,3]
[1,] 9.040835e-01 0.01710946 0.0000000
[2,] 9.589041e-02 0.93539237 0.1124910
[3,] 2.609263e-05 0.04749817 0.8875090
```

```
R> measureMRI(y, truth, tc.mcmcsub.prop2$prob)
```

```
$mse
[1] 0.0256319
$misclass
[1] 0.0873813
$rseVolume
[1] 0.05758643 0.06288223 0.05886432
$DSM
[1] 0.9245194 0.9120727 0.9077630
$conTable
          [,1]       [,2]      [,3]
[1,] 0.8978995 0.01541116 0.0000000
[2,] 0.1021005 0.94074924 0.1189545
[3,] 0.0000000 0.04383960 0.8810455
```

For function `initProp`, reasonably good initial guesses provide similar results as that from `initOtsu`. For data sets we studied, it is typical that different initialization procedures provide similar results given proper initial guess for `initProp`.

5. Discussion

In this paper, we introduced the R package **mritc** for MRI tissue classification. See Feng *et al.* (2012) for more details on the comparisons among different classification methods. Since the number of components is flexible, the methods introduced are not restricted to three types of tissues. Furthermore, the approaches are also suitable for classification of other imaging data, for example histological slices (Mete1 *et al.* 2007). We may incorporate more methods, for example in Jiang *et al.* (2010), into the package in the future.

Acknowledgments

We thank Dr. Vincent Magnotta for helpful discussions and the associate editor and two referees for their comments leading to the improvement of this paper. Research was supported in part by National Science Foundation grants DMS 06-04593 and 09-06398.

References

Besag J (1974). "Spatial Interaction and the Statistical Analysis of Lattice Systems." *Journal of the Royal Statistical Society B*, **36**(2), 192–236.

Besag J (1986). "On the Statistical Analysis of Dirty Pictures." *Journal of the Royal Statistical Society B*, **48**(3), 259–302.

Bordier C, Dojat M, Lafaye de Micheaux P (2011). "Temporal and Spatial Independent Component Analysis for fMRI Data Sets Embedded in the **AnalyzeFMRI** R Package." **44**(9), 1–24. URL http://www.jstatsoft.org/v44/i09/.

Bourgine P, Nadal JP (eds.) (2004). *Cognitive Economics: An Interdisciplinary Approach*. Springer-Verlag.

Celeux G, Hurn M, Robert CP (2000). "Computational and Inferential Difficulties with Mixture Posterior Distributions." *Journal of the American Statistical Association*, **95**(451), 957–970.

Chandra R, Menon R, Dagum L, Kohr D, Maydan D, McDonald J (2001). *Parallel Programming in OpenMP*. Morgan Kaufmann Publishers.

Collins DL, Zijdenbos AP, Kollokian V, Sled JG, Kabani NJ, Holmes CJ, Evans AC (1998). "Design and Construction of a Realistic Digital Brain Phantom." *IEEE Transactions on Medical Imaging*, **17**(3), 463–468.

Cuadra MB, Cammoun L, Butz T, Cuisenaire O, Thiran JP (2005). "Comparison and Validation of Tissue Modelization and Statistical Classification Methods in T1-Weighted MR Brain Images." *IEEE Transactions on Medical Imaging*, **24**(12), 1548–1565.

Dempster AP, Laird NM, Rubin DB (1977). "Maximum Likelihood from Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society B*, **39**(1), 1–38.

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

Feng D, Tierney L (2008). "Computing and Displaying Isosurfaces in R." *Journal of Statistical Software*, **28**(1). URL http://www.jstatsoft.org/v28/i01/.

Feng D, Tierney L, Magnotta V (2012). "MRI Tissue Classification Using High Resolution Bayesian Hidden Markov Normal Mixture Models." *Journal of the American Statistical Association*. Forthcoming.

Hauert C, Szabó G (2005). "Game Theory and Physics." *American Journal of Physics*, **73**(5), 405–414.

Hornak JP (2011). *The Basics of MRI*. URL http://www.cis.rit.edu/htbooks/mri.

Ising E (1925). "Beitrag zur Theorie des Ferromagnetismus." *Zeitschrift für Physik*, **31**, 253–258.

Jiang T, Navab N, Pluim JPW, Viergever MA (eds.) (2010). *Proceedings of Medical Image Computing and Computer-Assisted Intervention – MICCAI 2010*. Springer-Verlag.

Kohring GA (1996). "Ising Models of Social Impact: The Role of Cumulative Advantage." *Journal de Physique I*, **6**(2), 301–308.

Liao PS, Chen TS, Chung PC (2001). "A Fast Algorithm for Multilevel Thresholding." *Journal of Information Science and Engineering*, **17**, 713–727.

MacKay DJC (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

Marchini JL, Lafaye de Micheaux P (2009). *AnalyzeFMRI: Functions for Analysis of fMRI Datasets Stored in the ANALYZE or NIFTI Format*. R package version 1.1-11, URL http://CRAN.R-project.org/package=AnalyzeFMRI.

Mete1 M, Xu X, Fan CY, Shafirstein G (2007). "Automatic Delineation of Malignancy in Histopathological Head and Neck Slides." *BMC Bioinformatics*, **8**(Suppl 7), S17.

Mitchell T (1997). *Machine Learning*. McGraw Hill.

Otsu N (1979). "A Threshold Selection Method from Gray-Level Histograms." *IEEE Transactions on Systems, Man and Cybernetics*, **9**, 62–66.

Pham DL, Xu C, Prince JL (2000). "Current Methods in Medical Image Segmentation." *Annual Review of Biomedical Engineering*, **2**, 315–337.

Potts RB (1953). "Some Generalized Order-Disorder Transformations." In *Cambridge Philosophic Society*, volume 48, pp. 106–109.

Tabelow K, Polzehl J (2011). "Statistical Parametric Maps for Functional MRI Experiments in R: The Package **fmri**." *Journal of Statistical Software*, **44**(11), 1–21. URL http://www.jstatsoft.org/v44/i11/.

Tang PTP (1991). "Table Lookup Algorithms for Elementary Functions and Their Error Analysis." In *Proceedings of the 10th IEEE Symposium on Computer Arithmetic*, pp. 232–236.

Tierney L (2007). "A Brief Introduction to OpenMP." Department of Statistics & Actuarial Science, University of Iowa, URL http://www.stat.uiowa.edu/~luke/classes/295-hpc/notes/openmp.pdf.

Van Leemput K, Maes F, Vandermeulen D, Suetens P (2003). "A Unifying Framework for Partial Volume Segmentation of Brain MR Images." *IEEE Transactions on Medical Imaging*, **22**(1), 105–119.

Wilkinson DJ (2005). "Parallel Bayesian Computation." In EJ Kontoghiorghes (ed.), *Handbook of Parallel Computing and Statistics*, pp. 481–512. Marcel Dekker/CRC Press.

Winkler G (2003). *Image Analysis, Random Fields and Dynamic Monte Carlo Methods: A Mathematical Introduction*. 2nd edition. Springer-Verlag.

Zhang Y, Brady M, Smith S (2001). "Segmentation of Brain MR Images through a Hidden Markov Random Field Model and the Expectation-Maximization Algorithm." *IEEE Transactions on Medical Imaging*, **20**(1), 45–57.

Affiliation:

Dai Feng
Biometrics Research Department, Merck Research Lab
Merck & Co., Inc.
RY 33-300 P.O. Box 2000
Rahway, NJ 07065, United States of America
E-mail: dai_feng@merck.com

Luke Tierney
Department of Statistics & Actuarial Science
University of Iowa
Iowa City, IA 52242, United States of America
E-mail: luke-tierney@uiowa.edu

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 44, Issue 7 *Submitted:* 2010-10-28
October 2011 *Accepted:* 2011-06-20