

## Package FDRhmrF Version 1

By **Hai Shu** and **Bin Nan**

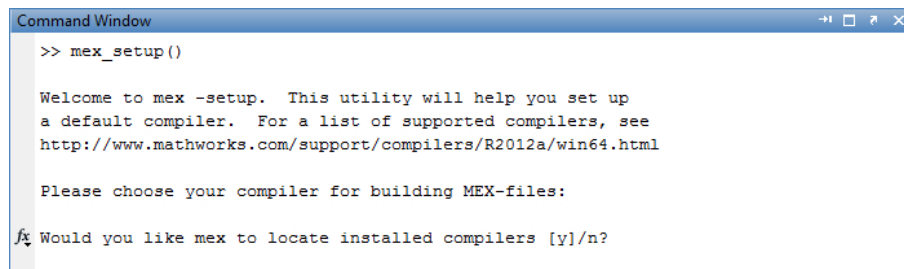
### 1. Introduction

The package `FDRhmrF` provides the implementation of multiple testing for three-dimensional (3D) image data using the false discovery rate (FDR) controlling procedure based on hidden Markov random field (HMRF) and local index of significance (LIS) (Shu and Nan, 2014). The package is coded in C++ source code, and called from MATLAB by a user-friendly interface.

### 2. Installation

The C++ source code is called by MEX-files from MATLAB, and thus both MATLAB and a C++ compiler are required before the installation of `FDRhmrF`. Here we only illustrate the installation on Windows (64-bit). The installation on other platforms should be similar.

Set the package folder as your current directory in MATLAB. To start the installation, at the MATLAB prompt `>>`, type: `mex_setup()`. The command window now should be:



```

Command Window
>> mex_setup()

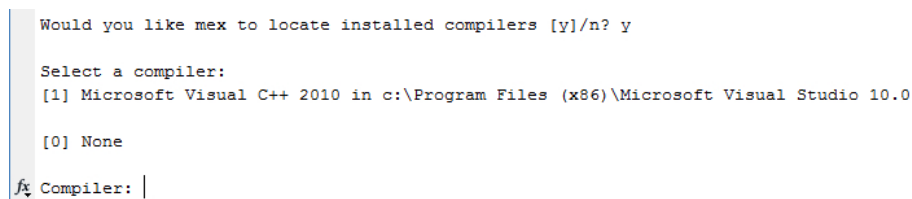
Welcome to mex -setup. This utility will help you set up
a default compiler. For a list of supported compilers, see
http://www.mathworks.com/support/compilers/R2012a/win64.html

Please choose your compiler for building MEX-files:

fx Would you like mex to locate installed compilers [y]/n?

```

Type `y`, then you will be asked to select a compiler:



```

Would you like mex to locate installed compilers [y]/n? y

Select a compiler:
[1] Microsoft Visual C++ 2010 in c:\Program Files (x86)\Microsoft Visual Studio 10.0

[0] None

fx Compiler: |

```

Then type the number of an available compiler:

```

Compiler: 1

Please verify your choices:

Compiler: Microsoft Visual C++ 2010
Location: c:\Program Files (x86)\Microsoft Visual Studio 10.0

fx Are these correct [y]/n?

```

Type y to confirm your selection, and then wait until the MATLAB prompt >> appears:

```

Are these correct [y]/n? y

*****
Warning: MEX-files generated using Microsoft Visual C++ 2010 require
that Microsoft Visual Studio 2010 run-time libraries be
available on the computer they are run on.
If you plan to redistribute your MEX-files to other MATLAB
users, be sure that they have the run-time libraries.
*****

Trying to update options file: C:\Users\Hai Shu\AppData\Roaming\MathWorks\MATLAB\R2012a\mexopts.bat
From template:           C:\PROGRA~1\MATLAB\R2012a\bin\win64\mexopts\msvc100opts.bat

Done . . .

*****
Warning: The MATLAB C and Fortran API has changed to support MATLAB
variables with more than 2^32-1 elements. In the near future
you will be required to update your code to utilize the new
API. You can find more information about this at:
http://www.mathworks.com/help/techdoc/matlab\_external/bsfinue-1.html
Building with the -largeArrayDims option enables the new API.
*****

fx >>

```

If there is no error message displayed, the installation is successfully finished and three MEX-files, `computelis.mexw64`, `fdrBH95.mexw64` and `fdrplis.mexw64`, should be now in your current directory.

### 3. Functions

#### 3.1 `fdrBH95`

##### Syntax

```
result=fdrBH95(input_info,reglocafile,pvaluefile,output_signal)
```

##### Description

`fdrBH95` is used to implement the FDR procedure of Benjamini and Hochberg (BH; 1995) for multiple testing. It outputs the signals of the brain region of interest (ROI) that are identified by the BH procedure into a file named `output_signal`. In the signal file, 1 indicates signal and 0 indicates non-signal. The returning vector `result = (r, n, threshold)`, where  $r$  is the number of discovered signals (rejections),  $n$  is the number of ROI's voxels (hypotheses), and  $threshold$  is the threshold of  $p$ -values used to identify signals (reject the null hypothesis).

### Arguments

`input_info`: a row vector =  $(Lx, Ly, Lz, q)$ , where  $Lx, Ly, Lz$  are the dimensions of the image lattice (including the brain part and the non-brain part) on  $x, y, z$  axes respectively and  $q$  is the FDR controlling level.

`reglocafile`: a string contains the name of the existing file storing the vectorized location of the ROI on the image lattice. You may use the MATLAB built-in function `reshape` to transform the 3D location  $(x, y, z)$  into the vectorized location  $i$ , where  $1 \leq x \leq Lx, 1 \leq y \leq Ly, 1 \leq z \leq Lz$  and  $1 \leq i \leq Lx \times Ly \times Lz$ .

`pvaluefile`: a string contains the name of the existing file storing the  $p$ -values on the whole image lattice including the brain part and the non-brain part.

`output_signal`: a string contains the name of the output file to store the result of the ROI's signals that are identified by the BH procedure. In the signal file, 1 indicates signal and 0 indicates non-signal.

### Example

Type:

```
input_info=[160,160,96,0.1];
reglocafile='loca_56.txt';
pvaluefile='pvalue_1vs2.txt';
```

```
output_signal='signals_BH_56.txt';
result=fdrBH95(input_info,reglocafile,pvaluefile,output_signal)
```

The returning result will be:

```
result =
0 920.0000 0.0001
```

### 3.2 *initialvalue*

#### **Syntax**

```
initials=initialvalue(signalfile,zvaluefile,reglocafile,L)
```

#### **Description**

*initialvalue* returns the initial values for the HMRF parameters including  $\beta, h$  of the Ising model and  $\mu_i, \sigma_i^2, p_i, i = 1, 2, \dots, L$ , of the non-null normal mixture. The elements of the returning vector *initials* are in the order of  $\beta, h, \mu_1, \sigma_1^2, p_1, \dots, \mu_L, \sigma_L^2, p_L$ .

#### **Arguments**

*signalfile*: a string contains the name of the existing file storing the ROI's signals that are usually identified by a simple multiple testing procedure, e.g., the BH procedure. In the signal file, 1 indicates signal and 0 indicates non-signal.

*zvaluefile*: a string contains the name of the existing file storing the  $z$ -values on the whole image lattice including the brain part and the non-brain part.

*reglocafile*: a string contains the name of the existing file storing the vectorized location of the ROI on the image lattice.

*L*: an integer indicates the number of the components in the non-null normal mixture.

#### **Example**

Type:

```
signalfile='signals_BH_56.txt';
zvaluefile='zvalue_1vs2.txt';
```

```
reglocafile='loca_56.txt';
L=2;
initials=initialvalue(signalfile,zvaluefile,reglocafile,L)
```

Then the returning result will be:

```
initials =
Columns 1 through 6
0 0 2.0000 1.0000 0.5000 -2.0000
Columns 7 through 8
1.0000 0.5000
```

### 3.3 *computelis*

#### Syntax

```
computelis(input_info,initials,zvaluefile,reglocafile,
           output_estimate,output_LIS)
computelis(input_info,initials,zvaluefile,reglocafile,
           output_estimate,output_LIS,output_process)
```

#### Description

`computelis` is used to run the generalized expectation-maximization algorithm (GEM; Shu and Nan 2014) to obtain the estimates of the HMRF parameters, and also to compute the LIS statistics of the ROI. The HMRF parameter estimates will be saved into the file `output_estimate`, and the ROI's LIS will be outputted into the file `output_LIS`. The output file `output_process` is optional to store the tracing information of the algorithm.

#### Arguments

`input_info`: a row vector =  $(Lx, Ly, Lz, L, sweep\_b = 1000, sweep\_r = 5000, sweep\_lis = 1e6, iter\_max = 5000)$ .  $Lx, Ly, Lz$  are the dimensions of the image lattice (including the brain

part and the non-brain part) on  $x, y, z$  axes respectively.  $L$  is the number of components in the non-null normal mixture. The last four elements are optional. In the GEM, *sweep\_b* is the number of iterations for the burn-in period of the Gibbs sampler, *sweep\_r* is the Gibbs-sampler sample size and *iter\_max* is the maximum number of GEM iterations. After running the GEM, the LIS statistics are computed by the Gibbs-sampler sample of size *sweep\_lis*. (*sweep\_b, sweep\_r, sweep\_lis, iter\_max*) is set to be (1000, 5000, 1e6, 5000) by default.

*initials*: a row vector contains the initial values for the HMRF parameters including  $\beta, h$  of the Ising model and  $\mu_i, \sigma_i^2, p_i, i = 1, 2, \dots, L$ , of the non-null normal mixture. The elements of this input vector *initials* should be in the order of  $\beta, h, \mu_1, \sigma_1^2, p_1, \dots, \mu_L, \sigma_L^2, p_L$ .

*zvaluefile*: a string contains the name of the existing file storing the  $z$ -values on the whole image lattice including the brain part and the non-brain part.

*reglocafile*: a string contains the name of the existing file storing the vectorized location of the ROI on the image lattice.

*output\_estimate*: a string contains the name of the output file to store the estimates of the HMRF parameters that are in the same order of the elements of *initials*.

*output\_LIS*: a string contains the name of the output file to store the LIS statistics.

*output\_process*: a string contains the name of the output file to store the tracing information of the algorithm. The tracing information includes the HMRF parameter estimates and the values of score function and information matrix obtained from each GEM iteration, and the time used to run the function `computelis`. *output\_process* is an optional input argument. The notation used in the output file *output\_process* is listed as follows:

`mul_hat[i], sigma1_sq_hat[i], pEll_hat[i], i=0, 1, \dots, L-1`: the mean, variance and proportion of the (i+1)-th component in the non-null normal mixture;

`U[i], i=0, 1`: the (i+1)-th element of the score function  $\mathbf{U}^{(t+1)}(\boldsymbol{\varphi})$ ;

`H_cond_mean[i], i=0, 1`: the (i+1)-th elements of  $E_{\boldsymbol{\Phi}^{(t)}}[\mathbf{H}(\boldsymbol{\Theta})|\mathbf{x}]$ ;

$H\_mean[i], i=0, 1$ : the  $(i+1)$ -th elements of  $E_{\varphi}[\mathbf{H}(\Theta)]$ ;

$I[i], i=0, 1, 2$ :  $I[0], I[2]$  are the diagonal entries, and  $I[1]$  is the off-diagonal entry of the information matrix  $\mathbf{I}(\varphi)$ ;

$lambda$ : the proportion of the regular Newton step, i.e.,  $\lambda_m$ ;

$delta\_Q2$ : the increase in  $Q_2$ -function;

$d1$ : the scaled distance between previous and current estimates, i.e.,

$$\max_i \left( \frac{|\Phi_i^{(t+1)} - \Phi_i^{(t)}|}{|\Phi_i^{(t)} + \epsilon_1|} \right).$$

### Example

Type:

```
input_info=[160,160,96,2,1000,5000,1e6,5000];
initials=[0,0,2,1,0.5,-2,1,0.5];
zvaluefile='zvalue_1vs2.txt';
reglocafile='loca_56.txt';
output_estimate='estimate_result.txt';
output_LIS='LIS_56.txt';
output_process='process_output.txt';
computelis(input_info,initials,zvaluefile,reglocafile,
           output_estimate,output_LIS,output_process)
```

### 3.4 *fdrplis*

#### Syntax

```
result=fdrplis(input_info,reg_no,reglocafile,LISfile,
               output_signal,output_LIS)
```

#### Description

`fdrplis` is used to implement the pooled LIS procedure (PLIS; Wei et al. 2009) for multiple testing. It outputs the signals and LIS statistics of the ROIs into the files `output_signal` and `output_LIS`, respectively. The returning vector `result=(r, n, threshold)`, where  $r$  is the number of discovered signals (rejections),  $n$  is the number of ROI's voxels (hypotheses), and  $threshold$  is the threshold of LIS statistics used to identify signals (reject the null hypothesis).

### Arguments

`input_info`: a row vector =  $(Lx, Ly, Lz, q)$ , where  $Lx, Ly, Lz$  are the dimensions of the image lattice (including the brain part and the non-brain part) on  $x, y, z$  axes respectively and  $q$  is the FDR controlling level.

`reg_no`: a row vector contains the number of each ROI.

`reglocafile`: a string contains the common part in the names of the files respectively storing each ROI's vectorized location on the image lattice. E.g., `reglocafile='loca'` if the files are `'loca_40.txt'` and `'loca_56.txt'`. The file format must be `".txt"`.

`LISfile`: a string contains the common part in the names of the files respectively storing each ROI's LIS statistics. E.g., `LISfile='LIS_'` if the files are `'LIS_40.txt'` and `'LIS_56.txt'`. The file format must be `".txt"`.

`output_signal`: a string contains the name of the output file to store the result of the ROIs' signals that are identified by the PLIS procedure. In the signal file, 1 indicates signal and 0 indicates non-signal. The file will contain a vector of size  $Lx \times Ly \times Lz$ , where the signal result of each ROI can be located by the ROI's vectorized location and the elements at the non-ROI locations are set to be 0.

`output_LIS`: a string contains the name of the output file to store the ROIs' LIS statistics. The file will contain a vector of size  $Lx \times Ly \times Lz$ , where the LIS statistics of each ROI can be located by the ROI's vectorized location and the elements at the non-ROI locations are



set to be 0.

### Example

Type:

```
input_info=[160,160,96,1e-3];
reg_no=[40,56];
reglocafile='loca_';
LISfile='LIS_';
output_signal='signals_plis_40&56.txt';
output_LIS='LIS_40&56.txt';
result=fdrplis(input_info,reg_no,reglocafile,LISfile,
               output_signal,output_LIS)
```

The returning result will be:

```
result =
1.0e+03 *
1.2690000000000000 2.5720000000000000 0.0000059670000000
```

## 4. Steps to implement the HMRF-based PLIS procedure

In this section, we show detailed steps to implement the HMRF-based PLIS procedure (Shu and Nan 2014) by the `FDRhmr` package. Example code is available in the MATLAB file `example.m`. Assuming there are  $n_1$  and  $n_2$  3D images for groups 1 and 2 on  $L_x \times L_y \times L_z$  image lattice respectively, the steps to do the pooled FDR analysis of ROIs for the comparison between groups 1 and 2 are as follows:

(1) Do a two-sample Welch's t-test (Welch, 1947) for each ROIs' voxel, and save the  $p$ -values and the associated  $z$ -values respectively into a 3D array of size  $L_x \times L_y \times L_z$  by the corresponding 3D voxels' locations. Transform the 3D arrays into vectors by using the MATLAB built-in function `reshape`. E.g., `pvalue_1d=reshape(pvalue_3d,1,Lx*Ly*Lz)` or

`pvalue_1d=reshape(pvalue_3d, Lx*Ly*Lz, 1)`. Similarly, obtain the vectorized location of each ROI.

(2) Use function `fdrBH95` to obtain each ROI's signal file, and then use `initialvalue` to obtain initial values of HMRF parameters for each ROI.

(3) Use function `computelis` to obtain each ROI's LIS statistics, then carry out the PLIS procedure using function `fdrplis`.

(4) `fdrplis` returns ROIs' signals and LIS statistics respectively in a vector of size  $Lx \times Ly \times Lz$ . You can transform the vectors back to 3D arrays by using the MATLAB built-in function `reshape`. E.g., `signal_3d=reshape(signal_1d, Lx, Ly, Lz)`.

## Acknowledgement

We thank Dr. Agner Fog (Technical University of Denmark) for his open C++ libraries of random number generators (Fog 2010; available at [www.agner.org/random](http://www.agner.org/random)), where the C++ files of `FDRhmrp`, `randomc.h`, `stocc.h`, `stocl.cpp`, `mersenne.cpp` and `userintf.cpp` are obtained.

## References

- Benjamini, Y., and Hochberg, Y. (1995). Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society, Series B* **57**, 289-300.
- Fog, A. (2010). Instructions for the random number generator libraries on [www.agner.org](http://www.agner.org). Available at [www.agner.org/random](http://www.agner.org/random), and also available in the folder of package `FDRhmrp`.
- Shu, H., and Nan, B. (2014). Multiple testing for neuroimaging via hidden Markov random field. Submitted to *Biometrics*.
- Wei, Z., Sun, W., Wang, K., and Hakonarson, H. (2009). Multiple testing in genome-wide association studies via hidden Markov models. *Bioinformatics* **25**, 2802-2808.

Welch, B. L. (1947). The generalization of 'Student's' problem when several different population variances are involved. *Biometrika* **34**, 28-35.