BHMSMAfMRI User Manual

Nilotpal Sanyal
Bayesian and Interdisciplinary Research Unit
Indian Statistical Institute, Calcutta, India
nsanyal_v@isical.ac.in

Welcome to BHMSMAfMRI, an R package to analyze functional MRI (fMRI) data. This manual shows how to systematically use the BHMSMAfMRI package functions to analyze fMRI data and should be helpful for the first-time user. In Section 1, we give a short introduction and overview of the methodology, and in Section 2, we discuss the package functions in a systematic way and apply them to analyze a simulated fMRI dataset.

1 Introduction and overview

The package BHMSMAfMRI performs Bayesian hierarchical multi-subject multiscale analysis (BHMSMA) of fMRI data (Sanyal and Ferreira, 2012). Though fMRI data is generally 3D, until now, BHMSMAfMRI considers analysis of 2D slices only.

The main features and usefulness of the BHMSMA methodology are that this methodology

(a) takes into account both the temporal and the spatial information contained in the fMRI data, (b) performs multi-subject analysis and borrows strength across subjects for precise estimation of the brain activations, and provides a straightforward way to obtain group activation map, and (c) does not use Markov Chain Monte Carlo (MCMC) simulation and is fast.

We model the temporal variation present in the fMRI dataset through a general linear model and then consider discrete wavelet transform of the standardized regression coefficients to harness the spatial information. In the wavelet space, on each wavelet coefficient we put a mixture prior that is a combination of a Gaussian density and a point mass at zero. This prior specification takes into account the sparsity of the wavelet coefficients. For the mixture probabilities we consider a prior that depend on few hyperparameters. We develop empirical Bayes methodology to estimate the hyperparameters and carry out inference without using MCMC simulation and with

approximation of one-dimensional integrals only. The use of mixture prior borrows strength across subjects. Finally, the posterior mean of the regression coefficients are obtained by using the posterior mean of the wavelet coefficients in the inverse discrete wavelet transform. Further, we use the average over subjects of the posterior wavelet coefficients in the inverse discrete wavelet transform to obtain posterior group image of the regression coefficients. For posterior uncertainty assessment we develop a simulation-based method for the computation of the posterior variance of the regression coefficients.

The BHMSMAfMRI package fits our model to the fMRI data and provides the estimates of the hyperparameters of the BHMSMA methodology along with their standard error estimates, the posterior mean of the wavelet coefficients, the posterior mean of the regression coefficients, samples from the posterior distribution of the regression coefficients, and the posterior group image of the regression coefficients. The posterior samples, in turn, can be used to compute the estimates of posterior standard deviation and posterior probability maps. In addition, when the true signal is available, the package provides mean squared error (MSE).

2 Using the package functions with examples

In this section we illustrate the use of the package functions. We assume that prior to applying our methodology, the fMRI data have been preprocessed for necessary corrections like realignment or motion correction, slice-timing correction, coregistration with anatomical image and normalization. However, the data must not be spatially filtered before applying BHMSMA, because our approach is to include the spatial information into modeling instead of filtering it out. Preprocessing can be perfomed by using available softwares/packages like SPM (Friston et al., 2007), BrainVoyager (Goebel et al., 2006), AFNI (Cox, 1996), and FSL (Smith et al., 2004).

In the following subsections, we show the use of the package functions in a systematic way.

2.1 Reading fMRI Data

The function *read.fmridata* can read fMRI data file(s) stored in ANALYZE format (.img/.hdr files), NIFTI format (.img/.hdr files or .nii files) or AFNI format (.BRIK/.HEAD files) within a directory. The reading of the fMRI data files is done using R packages AnalyzeFMRI (Bordier et al., 2011) and fmri (Karsten Tabelow, 2011), which are loaded when BHMSMAfMRI package is loaded.

We have provided a simulated fMRI dataset within the BHMSMAfMRI package. For the purpose of illustration in this vignette, we consider this dataset. The simulated

dataset contains data from 3 subjects. We will be analyzing data from a single axial slice. So, for brevity, we have provided only data from that slice in the simulated dataset. For each subject, we consider a time series of 9 noisy images of the slice of interest with image dimension 32×32. The following code illustrates, how the function *read.fmridata* can be used to read the data files from this simulated dataset. The simulated datset is extracted in the directory 'fmridata' within the R temporary folder.

```
fpath <- system.file("extdata", package="BHMSMAfMRI")
untar(paste0(fpath,"/fmridata.tar"), exdir=tempdir())
Data <- array(dim=c(3,32,32,9))
for(subject in 1:3)
{
  directory <- paste0(tempdir(),"/fmridata","/s0",subject,"/")
  a <- read.fmridata(directory, format="Analyze", prefix=paste0("s0",subject,"_t"),
  nimages=9, dim.image=c(32,32,1))
  Data[subject,,,] <- a[,,1,]
}
dim(a)
#[1] 32 32  1  9
```

The above code reads all the data files for all subjects into a 4D array Data. For each subject, the data are generated by adding Gaussian random noise to the true regression coefficient image with activation in three regions. The positions of two activation regions are varied across subject. The underlying design is taken to be a block design. The true regression coefficient images and the design matrix are also included in the package and can be read as follows.

```
data(fmridata)
names(fmridata)
#[1] "grid"         "nsubject"     "TrueCoeff"    "DesignMatrix"
grid <- fmridata$grid
nsubject <- fmridata$nsubject
TrueCoeff <- fmridata$TrueCoeff
DesignMatrix <- fmridata$DesignMatrix
dim(TrueCoeff)
#[1]  3 32 32
dim(DesignMatrix)
#[1] 9 2
```

Specifically, now we have TrueCoeff, an array of dimension (3, 32, 32) containing the true regression coefficients, Data, an array of dimension (3, 32, 32, 9) containing time series of noisy observations for all the subjects and DesignMatrix, containing the design matrix used to generate the data. Note that, the R package neuRosim (Welvaert et al., 2011) can be used to generate fMRI data.

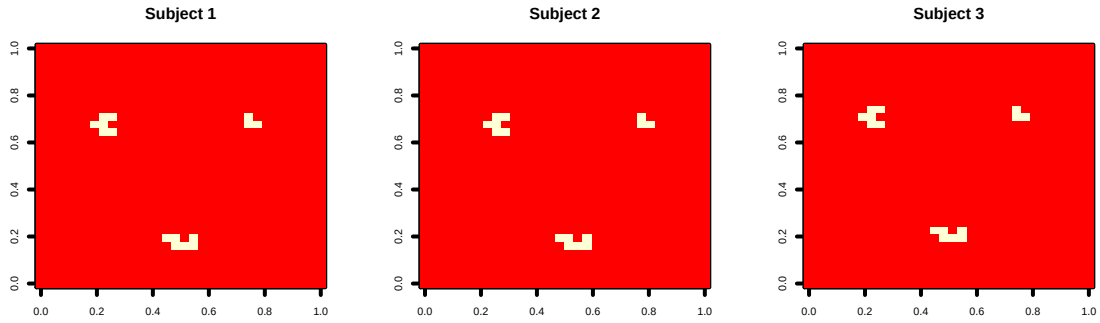

Figure 1: True regression coefficient images for the three subjects.

In Figure 1, let us look at the true regression coefficient images of the three subjects produced by the following code.

```
par(mfrow=c(1,3), cex=.2)
for(subject in 1:3)
image(TrueCoeff[subject,,], main=paste0("Subject ",subject))
```

2.2 Temporal modeling through GLM

Now, we fit a general linear model to the time series of each voxel and obtain the estimated regression coefficients by using the function *glmcoeff* as follows.

```
glmDat <- glmcoeff(nsubject, grid, Data, DesignMatrix)
names(glmDat)
#[1] "GLMCoeffStandardized" "GLMEstimatedSE"
dim(glmDat$GLMCoeffStandardized)
#[1]  3 32 32
dim(glmDat$GLMEstimatedSE)
#[1]  3 32 32
```

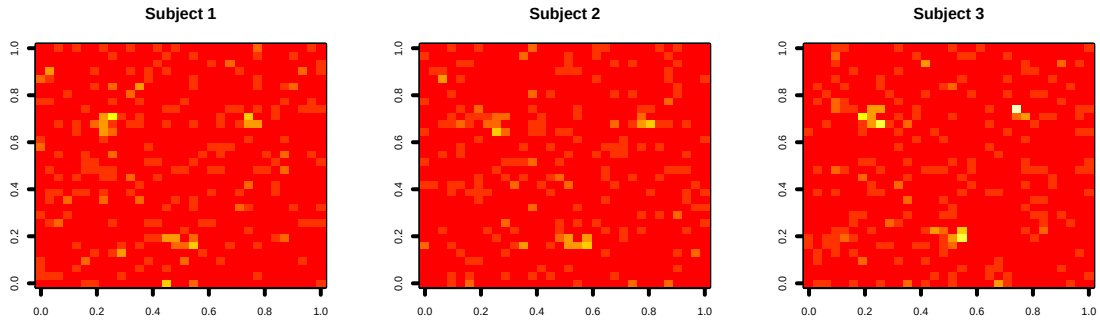

Figure 2: Estimated standardized regression coefficient images for the three subjects.

The output glmDat contains the estimated standardized regression coefficients and their standard error estimates. Figure 2, obtained by the following code, shows the images of the estimated standardized regression coefficients for the three subjects.

```
par(mfrow=c(1,3), cex=.2)
for(subject in 1:3)
image(abs(glmDat$GLMCoeffStandardized[subject,,]), col=heat.colors(8),zlim=c(0,max),
main=paste0("Subject ",subject))
```

2.3 Wavelet transform of the GLM coefficients

Next, we apply the discrete wavelet transform to the standardized regression coefficient images of each subject. The wavelet transformation is performed by the help of the R package wavethresh (Nason, 2013), which is loaded when BHMSMAfMRI package is loaded.

The function *waveletcoeff* outputs the wavelet coefficients for all the subjects within a matrix. The following code illustrates the use of the function *waveletcoeff* and the output.

```
wavecoeff.glmDat <- waveletcoeff(nsubject, grid, glmDat$GLMCoeffStandardized,
            wave.family="DaubLeAsymm", filter.number=6, bc="periodic")
names(wavecoeff.glmDat)
#[1] "WaveletCoefficientMatrix"
dim(wavecoeff.glmDat$WaveletCoefficientMatrix)
#[1]    3 1023
```

In the wavelet transform, the user can choose the wavelet family (one of "DaubLeAsymm" and "DaubExPhase"), the number of vanishing moments (filter.number) and the boundary condition ("symmetric" or "periodic") to be applied. For fMRI data, we recommend the use of Daubechies least asymmetric wavelet transform ("DaubLeAsymm") with 6 vanishing moments and periodic boundary condition.

2.4 Estimating the BHMSMA model hyperparameters

The BHMSMA model has six hyperparameters, which are estimated by their maximum likelihood estimates (MLEs) following an empirical Bayes approach. We can estimate the hyperparameters by performing multi-subject analysis or single subject analysis. In multi-subject analysis, the likelihood function of the hyperparameters is constructed over all subjects and maximized to obtain their estimates. In single subject analysis, for each subject, separate likelihood function of the hyperparameters is constructed and maximized. Hence, for single subject analysis, for each subject we obtain a set of estimates of the hyperparameters. Clearly, multi-subject analysis

benefits from being able to borrow strength across subjects and produces more precise estimates.

The function *hyperparamest* computes the hyperparameter estimates and their standard error estimates. The type of analysis must be specified as analysis="multi" or "single". The following code illustrates the use of the function *hyperparamest* and the output. The code may take some minutes to run.

```
hyparEst <- hyperparamest(nsubject, grid, wavecoeff.glmDat$WaveletCoefficientMatrix,
            analysis = "multi")
names(hyparEst)
#[1] "hyperparam"    "hyperparamVar"
round(hyparEst$hyperparam,3)
#[1] 1.181 1.337 0.431 0.675 3.663 0.201
hyparEst$hyperparamVar
#               [,1]          [,2]          [,3]          [,4]          [,5]          [,6]
#[1,]  1.307551e-19  5.682199e-35 -4.577911e-36 -1.433021e-35 -1.945261e-34 -6.416180e-36
#[2,]  5.682199e-35  1.676239e-19 -2.332483e-35 -8.518254e-35  4.184757e-34 -1.210776e-35
#[3,] -4.577911e-36 -2.332483e-35  1.740835e-20  5.228802e-36  7.097862e-35  2.341133e-36
#[4,] -1.433021e-35 -8.518254e-35  5.228802e-36  4.264501e-20 -8.887361e-35 -7.328437e-36
#[5,] -1.945261e-34  4.184757e-34  7.097862e-35 -8.887361e-35  1.257300e-18 -3.316008e-36
#[6,] -6.416180e-36 -1.210776e-35  2.341133e-36 -7.328437e-36 -3.316008e-36  3.799562e-21
```

From the hyperparameter estimates, we can compute the estimates of a_{kl}, b_{kl} and c_{kl} (Sanyal and Ferreira, 2012) for all levels as follows.

```
a.kl <- hyparEst$hyperparam[1] * 2^(-hyparEst$hyperparam[2] * (0:4))
b.kl <- hyparEst$hyperparam[3] * 2^(-hyparEst$hyperparam[4] * (0:4))
c.kl <- hyparEst$hyperparam[5] * 2^(-hyparEst$hyperparam[6] * (0:4))
round(a.kl,3)
#[1] 1.181 0.467 0.185 0.073 0.029
round(b.kl,3)
#[1] 0.431 0.270 0.169 0.106 0.066
round(c.kl,3)
#[1] 3.663 3.186 2.771 2.410 2.096
```

2.5 Computing posterior distribution of the wavelet coefficients

Given the values of the hyperparameters, the marginal posterior distribution of the wavelet coefficients is a mixture of a Gaussian and a point mass at zero with mixture probabilities $\bar{p}_{iklj}$. The BHMSMA methodology computes $\bar{p}_{iklj}$ values using Newton-Cotes numerical integration method. The function *pikljbar* computes the values $\bar{p}_{iklj}$ for all subjects and returns as a matrix. The type of analysis must be specified. The following code illustrates its use. The code may take a few minutes to run.

```
pklj.bar <- pikljbar(nsubject, grid, wavecoeff.glmDat$WaveletCoefficientMatrix,
            hyparEst$hyperparam, analysis = "multi")
names(pklj.bar)
#[1] "pklj.bar"
dim(pklj.bar$pklj.bar)
#[1]    3 1023
round(pklj.bar$pklj.bar[1,1:20],4)
# [1] 0.9958 0.9739 0.9835 0.8423 0.8931 0.8522 0.8961 0.8416 0.9996 0.9773 0.9742
#[12] 0.9270 0.9927 0.9806 0.9962 0.4423 0.7358 0.7764 0.2933 0.3426
```

Once $\bar{p}_{iklj}$ values are obtained, the marginal posterior distribution of the wavelet coefficients are entirely known. With the hyperparameter estimates and the $\bar{p}_{iklj}$ values, the function *postwaveletcoeff* computes the posterior mean and the posterior median of the wavelet coefficients. The type of analysis must be mentioned. The following code shows its use.

```
postwavelet <- postwaveletcoeff(nsubject, grid, wavecoeff.glmDat$WaveletCoefficientMatrix,
            hyparEst$hyperparam, pklj.bar = pklj.bar$pklj.bar,
            analysis = "multi")
names(postwavelet)
#[1] "PostMeanWaveletCoeff"   "PostMedianWaveletCoeff"
dim(postwavelet$PostMeanWaveletCoeff)
#[1]    3 1023
dim(postwavelet$PostMedianWaveletCoeff)
#[1]    3 1023
```

2.6 Computing posterior mean of the regression coefficients

Given the posterior mean of the wavelet coefficients, the function *postglmcoeff* can be used to obtain the posterior means of the regression coefficients. The following code shows its use.

```
postglm <- postglmcoeff(nsubject, grid, glmDat$GLMCoeffStandardized,
           postwavelet$PostMeanWaveletCoeff, wave.family="DaubLeAsymm",
             filter.number=6, bc="periodic")
names(postglm)
#[1] "GLMcoeffposterior"
dim(postglm$GLMcoeffposterior)
#[1]  3 32 32
```

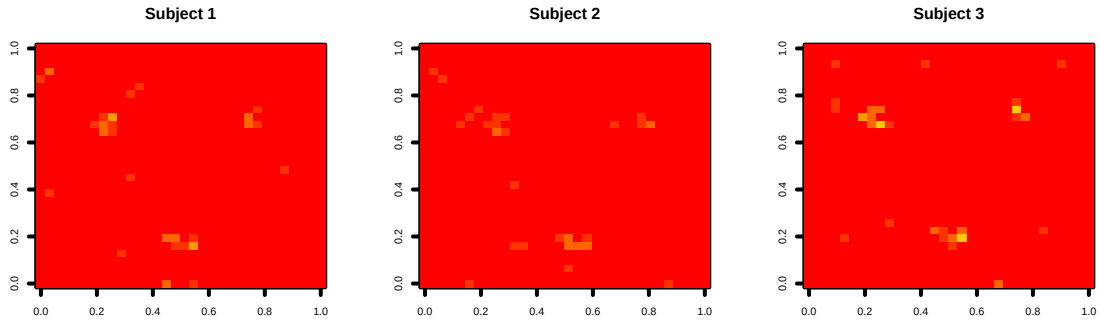

Figure 3: Posterior standardized regression coefficient images for the three subjects obtained by BHMSMA.

Figure 3, obtained by the following code, shows the images of the posterior standardized regression coefficients for the three subjects.

```
par(mfrow=c(1,3), cex=.2)
for(subject in 1:3)
image(abs(postglm$GLMcoeffposterior[subject,,]), col=heat.colors(8),zlim=c(0,max),
main=paste0("Subject ",subject))
```

As the true coefficients are known, we can compute the mean squared error (MSE) using the following code.

```
MSE <- c()
for (i in 1:nsubject)
 MSE[i] <- sum((as.vector(TrueCoeff[i, , ]/glmDat$GLMEstimatedSE[i,,])
                          - as.vector(postglm$GLMcoeffposterior[i,,]))^2)
round(MSE,3)
#[1] 301.127 257.521 287.048
```

In Sanyal and Ferreira (2012), we show that our multi-subject methodology performs better than some existing methodologies in terms of MSE.

2.7 The function *BHMSMA*

All the above stages beginning from obtaining the estimated regression coefficients by fitting GLM to obtaining the posterior regression coefficients are combined within the function *BHMSMA*. The use of the function *BHMSMA* and its output are shown in the code below. The type of analysis must be mentioned.

```
BHMSMAoutput.multi <- BHMSMA ( nsubject, grid, Data, DesignMatrix, TrueCoeff,
                      analysis="multi", saveplot=FALSE )
names(BHMSMAoutput.multi)
#[1] "GLMCoeffStandardized"     "GLMEstimatedSE"           "WaveletCoefficientMatrix"
#[4] "hyperparam"               "hyperparamVar"            "pklj.bar"
#[7] "PostMeanWaveletCoeff"    "GLMcoeffposterior"        "MSE"
```

Note that, when the true coefficients are available, the function *BHMSMA* returns the mean squared errors (MSEs) for all the subjects.

2.8 Posterior simulation and uncertainty estimation

In order to simulate observations from the posterior distribution of the regression coefficients, the function *postsamples* can be used. The type of analysis must be mentioned. The code below shows its use.

```
Postsamples <- postsamples( nsample=50, nsubject, grid, glmDat$GLMCoeffStandardized,
                wavecoeff.glmDat$WaveletCoefficientMatrix,  hyparEst$hyperparam,
                pklj.bar$pklj.bar, analysis="multi")
names(Postsamples)
#[1] "samples"       "postdiscovery"
dim(Postsamples$samples)
#[1]  3 32 32 50
dim(Postsamples$postdiscovery)
#[1]  3 32 32
```

The argument nsample denotes the number of samples to be drawn. We can see *postsamples* returns the posterior samples and the probabilities of posterior discovery (Morris et al., 2011) for all the subjects. Figure 4, obtained by the following code, shows the posterior discovery images based on the above 50 samples for the three subjects.

```
par(mfrow=c(1,3), cex=.2)
for(subject in 1:3)
image(Postsamples$postdiscovery[subject,,], main=paste0("Subject ",subject))
```

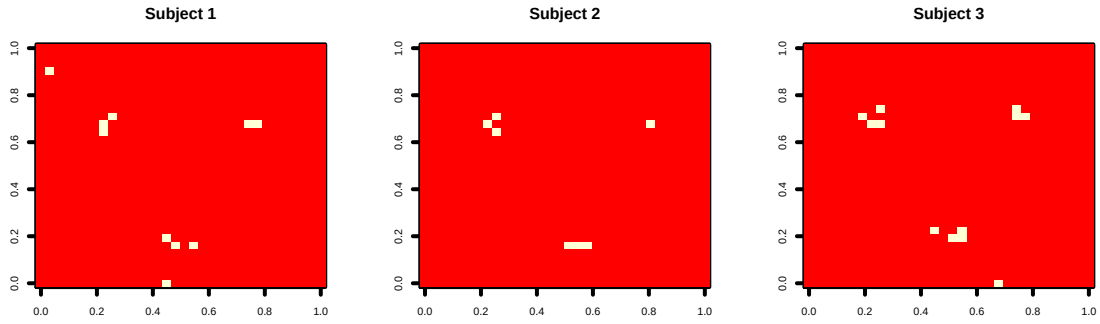

Figure 4: Posterior discovery images for the three subjects.

From the posterior samples, the posterior standard deviations of the regression coefficients can be computed as follows.

```
postsd <- array(dim=c(nsubject,grid,grid))
for(subject in 1:nsubject)
 postsd[subject,,] <- apply(Postsamples$samples[subject,,,], 1:2, sd)
round(postsd[1,1:5,1:5],3)
#      [,1]  [,2]  [,3]  [,4]  [,5]
#[1,] 0.560 0.454 0.526 0.732 0.553
#[2,] 0.736 0.704 0.565 0.722 0.623
#[3,] 0.914 0.589 0.311 0.410 0.489
#[4,] 0.735 0.572 0.493 0.505 0.560
#[5,] 0.815 0.598 0.759 0.640 0.696
```

2.9 Posterior group image

Posterior group coefficients can be obtained by using the function *postgroupcoeff* as follows.

```
postgroup <- postgroupcoeff( nsubject, grid, glmDat$GLMCoeffStandardized,
             postwavelet$PostMeanWaveletCoeff)
names(postgroup)
#[1] "groupcoeff"
dim(postgroup$groupcoeff)
#[1] 32 32
```

Figure 5, obtained by the following code, shows the posterior group coefficient image for the simulated dataset.

```
image(abs(postgroup$groupcoeff),col=heat.colors(8),zlim=c(0,max))
```

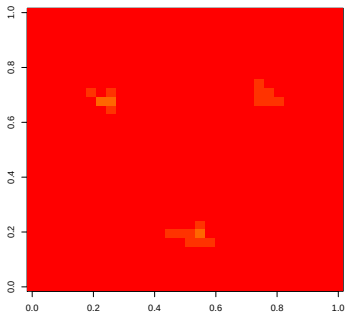

Figure 5: Posterior group regression coefficient image.

References

Bordier, C., Dojat, M., and de Micheaux, P. L. (2011). "Temporal and Spatial Independent Component Analysis for fMRI Data Sets Embedded in the AnalyzeFMRI R Package." *Journal of Statistical Software*, 44, 9, 1–24.

Cox, R. W. (1996). "AFNI: Software for analysis and visualization of functional magnetic resonance neuroimages." *Computers and Biomedical Research*, 29, 3, 162–173.

Friston, K., Ashburner, J., Kiebel, S., Nichols, T., and Penny, W., eds. (2007). *Statistical Parametric Mapping: The Analysis of Functional Brain Images*. Academic Press.

Goebel, R., Esposito, F., and Formisano, E. (2006). "Analysis of Functional Image Analysis Contest (FIAC) data with BrainVoyager QX: From single-subject to cortically aligned group General Linear Model analysis and self-organizing group Independent Component Analysis." *Human Brain Mapping*, 27, 5, 392–401.

Karsten Tabelow, J. P. (2011). "Statistical Parametric Maps for Functional MRI Experiments in R: The Package fmri." *Journal of Statistical Software*, 44, 11, 1–21.

Morris, J. S., Baladandayuthapani, V., Herrick, R. C., Sanna, P., and Gutstein, H. (2011). "Automated analysis of quantitative image data using isomorphic functional mixed models, with application to proteomic data." *Annals of Applied Statistics*, 5, 894–923.

Nason, G. (2013). *"wavethresh: Wavelets Statistics and Transforms."*. R package version 4.6.6.

Sanyal, N. and Ferreira, M. (2012). "Bayesian hierarchical multi-subject multiscale analysis of functional MRI data." *NeuroImage*, 63, 3, 1519–1531.

Smith, S., Jenkinson, M., Woolrich, M., Beckmann, C., Behrens, T., Johansen-Berg, H., Bannister, P., De Luca, M., Drobnjak, I., Flitney, D., Niazy, R., Saunders, J., Vickers, J., Zhang, Y., De Stefano, N., Brady, J., and Matthews, P. (2004). "Advances in functional and structural MR image analysis and implementation as FSL." *NeuroImage*, 23, SUPPL. 1, S208–S219.

Welvaert, M., Durnez, J., Moerkerke, B., Verdoolaege, G., and Rosseel, Y. (2011). "neuRosim: An R Package for Generating fMRI Data." *Journal of Statistical Software*, 44, 10, 1–18.