A Tutorial on Modeling fMRI Data using a General Linear Model

Natalie Koh[1] Adel Lee[2], Katie McLaughlin[3], Thomas Grabowski[1,4] and Tara M. Madhyastha[1]

[1]Department of Radiology, University of Washington, Seattle WA
[2]Etosha Business and Research Consulting, Mount Berry GA
[3]Department of Psychology, University of Washington, Seattle WA
[4]Department of Neurology, University of Washington, Seattle WA

Introduction

Scientists from many disciplines (including psychologists, neuroscientists, biostatisticians, engineers, and computer scientists) are interested in using, developing, and advancing neuroimaging methods. Functional magnetic resonance imaging (fMRI) has quickly become the most commonly used method for examining brain function. This paper describes a novel approach for performing statistical modeling of fMRI data. In our experience, there are often challenges in communicating statistical and analytical concepts across disciplines. A first step towards overcoming these communication challenges is to bridge the language and tools used by different disciplines. R is or is becoming the *lingua franca* of statistical analysis in diverse disciplines (Gentleman et al. 2004; Mizumoto and Plonsky 2016; Kamvar et al. 2017); therefore, there is didactic value to translating fMRI statistical analysis from the specialized software packages that are currently used to analyze fMRI data into R.

R is also helpful as a language for conducting analyses reproducibly (Gandrud 2016). As a scripting language, the R script itself can be self-documenting. Notably, R was used to drive a large scale collaborative replication of 100 psychological studies (Open Science Collaboration 2015).

However, an important reason to look towards fMRI analysis with R is that there is a serious disconnect between current approaches used used in psychology for data analysis and those that are implemented in specialized fMRI software, particularly in the domain of longitudinal models. Widely used software for analyzing task fMRI data rely predominantly on the general linear model (GLM) approach to analyze data (Poline and Brett 2012). The GLM is a broad class of models that assumes a linear relation between one (or more) dependent variables and one (or more) independent variables. In fMRI analysis, the GLM estimates the BOLD time series as a linear combination of several signal components and tests whether activity in a voxel is linked to any known input function or stimuli (Lindquist 2008).

This model has serious limitations, particularly in modeling longitudinal data, where a vast array of statistical models commonly used in psychology are simply not available for fMRI

analysis (T. Madhyastha et al. –). For example, numerous techniques for analyzing longitudinal developmental data are based in structural equation modeling (SEM) (Newsom 2015), which allows growth to be modeled not only as an outcome, but as a predictor of other processes, as in parallel process latent growth models (Cheong, MacKinnon, and Khoo 2003). Such a model could be used, for example, to examine whether *changes* in neural processes predict subsequent increases in cognitive symptoms. Neural changes across numerous time points could be examined as outcomes, predictors, correlates of other change processes, moderators, or mediators in SEM-based latent growth models. None of these models can be estimated in any of the widely used fMRI analysis software packages. However, these types of approaches are necessary for fMRI analysis to examine questions about longitudinal change such as "Does the amygdala become more active in a month when a person experiences more stress than is typical for them?" or "Are changes in brain networks in Parkinson disease compensatory to preserve cognitive function?". This need for more modeling flexibility to advance the study of developmental cognitive neuroscience, and similarly the trajectories of aging and neurodegenerative change, also motivates this tutorial as a first step to de-mystifying the scripting of more advanced analyses.

The purpose of this tutorial is to illustrate how statistical models commonly used for fMRI task analysis can be translated into the linear mixed effects modeling framework (Laird and Ware 1982). Specifically, we translate fMRI task analysis models as implemented in the FMRIB Software Library (FSL), a commonly used and representative specialized fMRI analysis package, to equivalent mixed effects model specifications that can be implemented using the nlme package (Pinheiro et al. 2017) in R (R Core Team 2014).

This tutorial is intended to help those with a background in the types of models used in behavioral analysis to understand how these models are applied to fMRI data using R as a bridging language. This tutorial is also written using Rmarkdown (see Supplemental Materials), so that it is itself an example for writing reproducible analysis.

fMRI Data

FMRI data is represented as a four-dimensional data file, which represents the blood oxygen level-dependent (BOLD) signal in the brain over time. An fMRI data file is a sequence of three-dimensional volumes. Each three-dimensional volume is an image of the brain, and the fourth dimension is time. The BOLD signal changes in the areas of the brain that support task demands, and the goal of fMRI task analysis is to identify these locations.

However, the variance in the fMRI signal reflects numerous physical influences besides activity-coupled hemodynamic change. Analysis of fMRI data is complex and involves removing these sources of noise to accurately measure the relatively small BOLD signal components. Tools to measure the BOLD signal were developed in the early 1990s. Briefly, these tools: a) remove sources of noise (e.g., from respiration and head motion); b) align the BOLD signal maps to each subject's brain anatomy; c) transform each subject's brain anatomy into a standardized space; and d) estimate the degree to which BOLD signal (i.e., neural activity) changes in response to task demands. These changes in response are

modeled at thousands of three-dimensional pixels, or ``voxels," in the brain. A test statistic is then calculated at each voxel location, resulting in multiple tests that need to be evaluated simultaneously. Finally, these multiple statistical tests are corrected to minimize false positives. These steps are usually performed using special-purpose software (e.g. FSL (Jenkinson et al. 2012), AFNI (Cox 1996) or MATLAB SPM (Friston et al. 2007).

We first describe and introduce some terminology for a basic fMRI experiment. A common fMRI experimental design is a "block design", where each subject is exposed to repeated "blocks" of different experimental conditions (e.g. series of experimental stimuli). One of these conditions is normally a baseline condition to which other conditions are compared. An important way to understand whether the brain is more or less active under different conditions is to examine a "contrast" between conditions. By examining contrasts, we can locate where the brain is more active under one condition than another.

One complexity in modeling the stimuli for different conditions is that the BOLD response is very slow. There is a delay of approximately 5 seconds between the time of the stimuli (i.e., when the neural response presumably occurs) and the corresponding BOLD signal response in the brain; the shape of this response is called the *hemodynamic response function* (Huettel, Song, and McCarthy 2014). To make the time course of the stimuli match the time course of the BOLD signal, the stimulus time course is typically convolved with this hemodynamic response function, so that it looks like what you would expect to see in the temporal BOLD signal. The convolved time-series for each stimulus is called an *explanatory variable*, or EV. These variables are used to explain the variability in the measured time-series data at each voxel location using statistical modeling. The EVs together, plus any covariates, form the *design matrix*, or the candidate variables that are expected to predict the BOLD signal. We will describe the form of this matrix later (see Figure 1 for an example) in more detail.

The changes in the BOLD signal that occur in a particular task are small compared to the spontaneous fluctuations in the BOLD signal in the brain. Therefore, we must present a condition enough times so that we have enough statistical power to detect a task-related effect. This often means increasing scan time, which is a problem because participants are less likely to be able to keep still for the entire duration of the scan. The solution is therefore to break very long scans into multiple task "runs" that occur during a single session inside a MRI scanner (typically lasting for a few minutes). Each participant may have multiple task runs. Ultimately, all runs are combined across individuals to estimate the effects (or contrasts) of interest in what we call a *group-level analysis*.

Preprocessing

Preprocessing refers to all the cleaning that is performed on fMRI data before any statistical analysis. Preprocessing is absolutely necessary to reduce and remove known sources of noise in fMRI data.

Most preprocessing steps can and should be accomplished with specialized software. Because the field and our understanding of the downstream effects of fMRI preprocessing

on subsequent statistical analysis are still evolving, there is no exact consensus on a standard preprocessing pipeline. It is important to note that the choice of a sequence of preprocessing steps, or "pipeline," can affect statistical results (Carp 2012; Bright and Murphy 2015). Moreover, processing pipelines that optimize the signal of interest may vary depending on the task being modeled, or even the population (Churchill et al. 2017).

For an introductory description of typical fMRI preprocessing steps, see (Huettel, Song, and McCarthy 2014; Poldrack, Mumford, and Nichols 2011). Most of these steps can be performed using a workflow that is scripted (e.g. using bash) or using some sort of workflow management system (Gorgolewski et al. 2011; Askren et al. 2016). This allows the interchange of preprocessing algorithms from the different major neuroimaging packages, and the separation of preprocessing from statistical modeling.

As part of this preprocessing workflow, several "nuisance regressors" are usually calculated. Nuisance regressors are covariates in the model that account for sources of noise caused by motion, respiration, changes in blood flow related to the cardiac cycle, and so on. There is usually one nuisance regressor per volume of the 4D fMRI image. Often one also includes the first derivatives of the EVs to account for slight differences in timing (e.g. if there are regional differences in the precise form of the hemodynamic response function).

The BOLD signal at any particular timepoint is significantly correlated with itself at different delays. This is called *autocorrelation*, and a failure to account for temporal autocorrelation in the residual errors of a model is likely to result in the underestimation of the standard errors of the model coefficients, and an increase in the number of false positives.

One technique commonly used to address autocorrelation in the BOLD signal is prewhitening. This important step is typically not included in preprocessing of task-related fMRI, but is instead performed at the same time as model fitting. This step is necessary to remove autocorrelation between voxel time series that biases parameter estimates in task analysis, but is not recommended for resting state analysis because it may remove the slow fluctuations that are of interest. A common prewhitening approach is to fit an autoregressive $AR(p)$ model to the residuals of the regression model to obtain estimates of the autoregressive parameters. These parameters can then be used to modify the design matrix, resulting in a regression model with uncorrelated errors.

Several studies have established that some prewhitening approaches are better than others (e.g. Woolrich et al. 2001). We highlight the importance of accounting for autocorrelation and describe a way of doing so within the mixed effects model framework, but evaluating this approach in comparison to existing implementations in dedicated fMRI analysis software is outside the scope of this tutorial.

The General Linear Model

The basic approach to analyzing fMRI data is to fit a linear model to the time series data measured at each voxel Y. The predictors of this model are the EVs that describe the

experimental events of interest, covariates that might account for some of the difference in the BOLD signal (such as age or sex, or the first derivatives of the EVs), and nuisance regressors that are of no interest, but must be controlled for to account for variance unrelated to experimental events of interest (such as the amount of motion or physiological noise). For a more in-depth description of assumptions and caveats of the GLM model in fMRI analysis, see (Monti 2011; Pernet 2014).

Task fMRI Example

Let us consider an example of an fMRI experiment where there are 4 conditions and one task run per subject.

The four conditions are:

- Fixation cross (EV_b): The subject sees a white fixation cross on a black screen and lets thoughts wander. This is the baseline condition.
- 0-Back (EV_1): The subject sees a series of letters, and presses a button when the letter is an "X."
- 1-Back (EV_2): The subject sees a series of letters, and presses a button when the letter is the same as the one that preceded it. This task engages working memory.
- 2-Back (EV_3): The subject sees a series of letters, and presses a button when the letter is the same as the one that was presented two letters ago. This task engages working memory and is more difficult than 1-Back.

The stimulus corresponding to each condition is either 1 or 0 to indicate whether or not the condition is presented at a given point in time. Each stimulus is then convolved with a hemodynamic response function to create the EV for each condition. Typically, null events (i.e. baseline, rest, or fixation periods during the task) are not explicitly modeled in the design matrix. This is because including an EV that corresponds to the baseline leads to multicollinearity, where at least one of the EVs in the design matrix can be predicted by a linear combination of the others with a high degree of accuracy (Pernet 2014). Such a model is considered to be over-parameterized, because there is no unique solution for the model parameters. Therefore, we often include the baseline condition implicitly in the model by including an intercept and a term for each of the non-baseline EV conditions (i.e. include a column of ones and EV_1, EV_2 and EV_3 in the design matrix). By including these terms in the model, we can ask scientific questions about the contrasts between the experimental conditions in our model. The list below is not exhaustive, but it provides an idea of the kinds of things we may want to know.

- 0-Back > baseline: Tells us when there is activity related to 0-Back that is more than just looking at a symbol on a screen.
- 1-Back > baseline: Tells us when there is activity related to 1-Back that is more than just looking at a symbol on a screen.
- 1-Back > 0-Back: Tells us when there is activity related to 1-Back that is more than just seeing a letter and reacting to it. In this case, the difference is probably related to the working memory component of the task.

- 1-Back > 0: Tells us when there is any activity related to the 1-Back time course. But it might just have to do with seeing a symbol on the screen.
- 2-Back > 1-Back: Tells us where the brain is more active in 2-Back than in 1-Back. The difference is probably related to the added difficulty of 2-Back.

Modeling in FSL

We briefly describe how this simple experiment is modeled in FSL. For more details, see FSL's documentation on FEAT and GLM analysis using FSL tools.

FSL uses a two-level model. The first level is the individual subject level. For each subject k, where $k = 1 \ldots N$, it models the BOLD signal at each voxel using the following equation:

$$Y_k = X_k \beta_k + e_k \qquad (1)$$

Where Y_k is a vector of length T capturing the BOLD time series observed at each voxel in the brain, X_k is the $T \times P_k$ design matrix that the experimenter specifies, β_k is a column vector of regression co-efficients for the EVs in the design matrix X_k, and e_k is a column vector of error terms. Note that in this example, we assume only one run per subject. We further assume that the first level errors $e_k \sim N(0, \sigma_k^2 V)$.[1] In FSL, the correlation matrix is typically used to model the autocorrelation in the error terms.

Next, the β_k coefficients estimated for each subject are used to fit a second level model,

$$\beta_k = X_{gk} \beta_g + e_{gk} \qquad (2)$$

Here, X_{gk} is the design matrix for the group, indicating, for example, the presence or absence of a disease for each subject. β_g represents the population-level effects for each of the EVs included in β_k. We assume that the error term $e_{gk} \sim N(0, R_g)$, where R_g is the variance-covariance matrix of the group-level errors.

In practice, when implementing this model in FSL, the user will specify the contrasts of interest for the analysis. A contrast is a linear combination of the parameter estimates (i.e. β values) of the EVs. Contrasts allow us to ask questions like: is the effect of 2-Back > 1-Back (for the population the study sample is drawn from)? Specifically, we are asking where in the brain the β coefficient for the EV corresponding to the 2-Back condition is greater than the β coefficient for the EV corresponding to the 1-Back condition.

If we assume that individual subjects are modeled at the first level and that the subjects' effects are combined at the second level, we study differences between the relevant group level β_g parameters to draw inference about the *population* that we sampled from. To test the statistical significance of these contrasts, FSL uses a Bayesian framework that derives the posterior distribution of the second level coefficients β_g. Woolrich et al. (2004) show

[1] This notation $N(0, \sigma_k^2 V)$ mean this is a normal distribution with mean of 0, variance of σ_k^2 and $T \times T$ correlation matrix V

that the posterior distribution of β_g depends on the data Y only through the first-level model summary statistics, and this increases computational speed substantially (compared to standard linear mixed effects model fitting procedures).

The FSL software works by "carrying up" only the summary statistic information corresponding to the contrast of interest to the second level. This includes summary statistics like the estimated mean, variance, and the degrees of freedom of the first level β_k coefficients.

Modeling in R: The Linear Mixed Effects Model

We now introduce the Laird and Ware (Laird and Ware 1982) formulation of the GLM model as used in the nlme package. Using matrix notation, the n_k observations of individual k is modeled as

$$Y_k = \beta_0 + X_{k1}\beta_1 + \cdots + X_{kp}\beta_p + b_{k0} + Z_{k1}b_{k1} + \cdots + Z_{kq}b_{kq} + e_k \qquad (3)$$

for $k = 1, \ldots, M$ individuals.

Here, Y_k is the $n_k \times 1$ column vector of observed values for individual k. The X_{k1},..., X_{kp} are the $n_k \times 1$ vectors for each of the p regressors included in the model. The regression coefficients β_0,..., β_p are the fixed-effect coefficients, and are identical across all individuals. The $n_k \times 1$ column vectors Z_{k1},..., Z_{kq} are the random effect regressors and the corresponding terms b_{k0}, b_{k1},...,b_{kq} are the random effects. Note that q is the total number of random effects included in the model. The random effects reflect the variability across subjects for each of the regressors Z_{k1},..., Z_{kq}, and are typically assumed to be normally distributed $(b_{k0}, \ldots, b_{kq}) \sim N(0, \psi)$, where ψ is a $(q+1)\times(q+1)$ variance-covariance matrix. The $n_k \times 1$ vector of within-group errors $e_k = (e_{k1}, \ldots, e_{kn_k})$ are also assumed to be normally distributed $N(0, \sigma^2 V)$, with variance σ^2 and correlation matrix V.

Using the lme function of the nlme package in R, we can fit the above model using maximum likelihood. For our purposes, we are interested in making inferences about a contrast $c'\beta$, where c is the contrast vector and $\beta = (\beta_0, \ldots, \beta_p)$. The function lme will give us a test statistic for our contrast, and allow us to calculate the corresponding p-value for that contrast of interest.

Translation Between FSL and Linear Mixed Effects Models

Here we illustrate how the FSL first-level and second-level models (equations (1) and (2)) translate into the mixed-effects model formulation (Section 4.3) for our example given above in Section 4.1.

For our example, the first level design matrix for subject k, X_k, has four columns. We assume that there are T volumes of fMRI data in the fMRI timeseries for each subject. For simplicity, we exclude nuisance variables in this example. If included, these would be additional columns that would represent volume-wise regressors for motion parameters, physiological signals, the first derivative of the EVs, and so on.

$$X_k = \begin{bmatrix} 1 & EV_{11} & EV_{21} & EV_{31} \\ 1 & EV_{12} & EV_{22} & EV_{32} \\ 1 & EV_{13} & EV_{23} & EV_{33} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & EV_{1T} & EV_{2T} & EV_{3T} \end{bmatrix} = [X_{k0} \quad X_{k1} \quad X_{k2} \quad X_{k3}] \qquad (4)$$

In this notation, X_{k0} is a column vector of ones for the intercept, X_{k1} is the column vector for EV_1, X_{k2} is the column vector for EV_2 and X_{k3} is the column vector for EV_3.

In FSL, this design matrix would be visualized as in Figure 1.

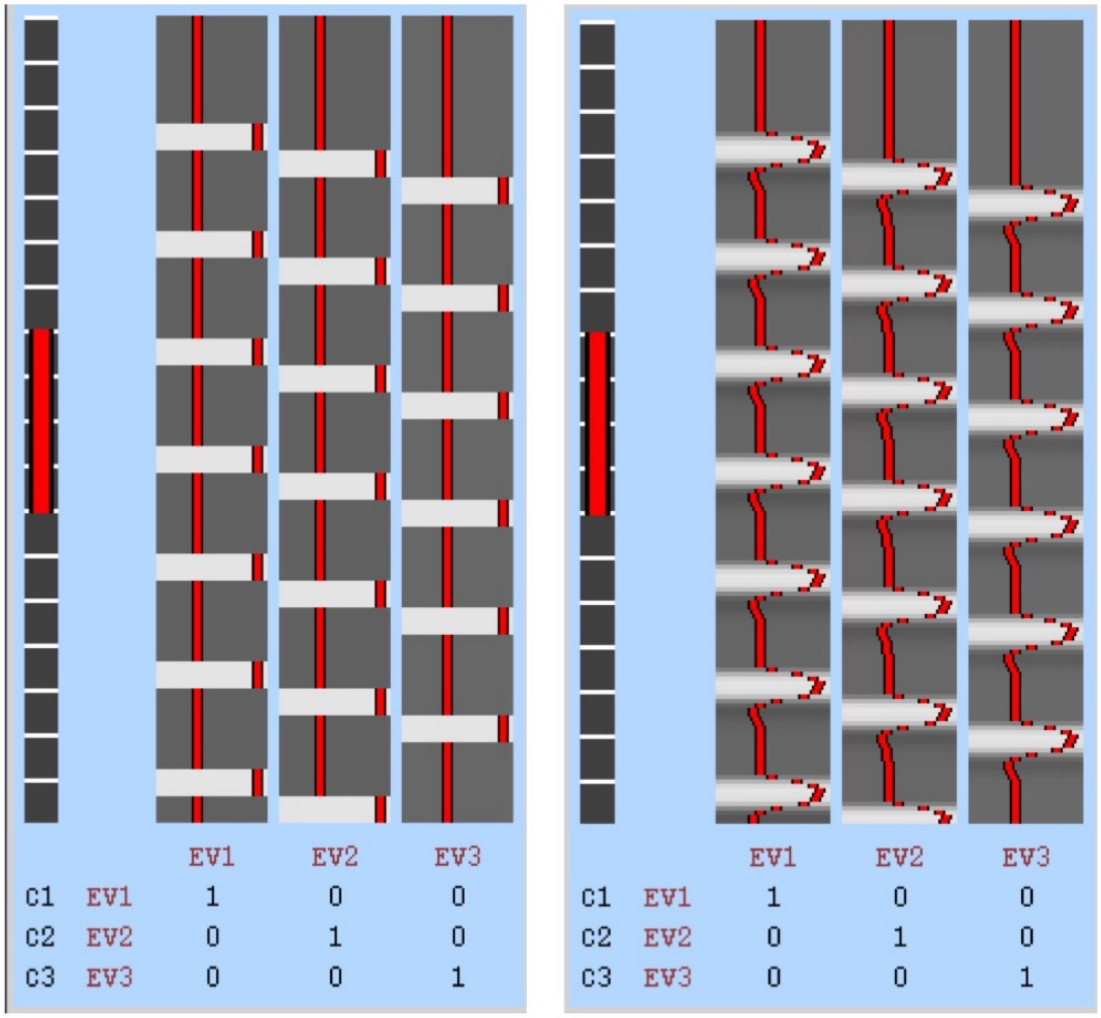


Figure 1 GLM model design for the n-back block task. On the left, the basic shapes of the stimuli functions for each of the EVs in modeled. The right shows the stimuli functions convolved with the double gamma hemodynamic response function. These EVs are used in the design matrix.

The corresponding vector of regression coefficients is

$$\beta_k = \begin{bmatrix} \beta_{k0} \\ \beta_{k1} \\ \beta_{k2} \\ \beta_{k3} \end{bmatrix} \qquad (5)$$

where β_{k0} is the intercept, β_{k1} is the regression coefficient corresponding to EV1, β_{k2} is the coefficient corresponding to EV_2, and β_{k3} is the coefficient corresponding to EV_3.

For the second-level model, assume that we are interested in estimating the average effect of each EV for the whole group (i.e., the "population effect").

Then,

$$X_g = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6)$$

To understand how the FSL two-level model specified above translates into a linear mixed effects model, as specified for nlme, we substitute the second-level model (2) into the first-level model (1):

$$\begin{aligned} Y_k &= X_k\beta_k + e_k \\ &= X_k[X_{gk}\beta_g + e_{gk}] + e_k \\ &= X_k X_{gk}\beta_g + X_k e_{gk} + e_k \\ &= [X_{k0} \quad X_{k1} \quad X_{k2} \quad X_{k3}] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \beta_{g0} \\ \beta_{g1} \\ \beta_{g2} \\ \beta_{g3} \end{bmatrix} + \\ &\quad [X_{k0} \quad X_{k1} \quad X_{k2} \quad X_{k3}] \begin{bmatrix} e_{gk0} \\ e_{gk1} \\ e_{gk2} \\ e_{gk3} \end{bmatrix} + e_k \\ &= X_{k0}\beta_{g0} + X_{k1}\beta_{g1} + X_{k2}\beta_{g2} + X_{k3}\beta_{g3} + X_{k0}e_{gk0} + X_{k1}e_{gk1} + X_{k2}e_{gk2} + X_{k3}e_{gk3} + e_k \end{aligned} \qquad (7)$$

for $e_k \sim N(0, \sigma_k^2 V)$ and $e_g \sim N(0, R_g)$.

We will refer to (7) as the translated FSL model. In this model, β_{g1}, β_{g2}, and β_{g3} are the population effects for EV_1, EV_2, and EV_3, respectively. These are referred to as the fixed effects in the linear mixed effects modeling framework. The group-level error terms e_{gk0}, e_{gk1}, e_{gk2}, and e_{gk3} reflect the random variability of the intercepts and slopes around their respective populations effects. These are the random effects.

Making a direct comparison between the translated FSL model (7) and the linear mixed effects model specified above (3), we see that β_{g0},... β_{g3} of the translated FSL model correspond to the fixed effects β_0,..., β_p (for $p = 3$) of the mixed effects model. Likewise, the group-level errors e_{gk0},..., e_{gk3} in the FSL/translated model are the b_{k0},..., b_{kq} in the mixed

effects model, and $q = 3$ (i.e., three random effects). Lastly, the random effect regressors Z_k specified in the mixed effects model are the columns of the design matrix X_k for the translated FSL model.

It is important to recognize that this translated FSL model (7) does not fully reflect the relationship between the two-level FSL model [(1) and (2)] and the mixed effect model (3), because the FSL model, as implemented in FEAT, assumes a Bayesian framework for inference. The linear mixed effects model, on the other hand, typically assumes a classical framework for inference.

R Tutorial

In this tutorial, we will simulate and then model the time series of a single voxel for n = 10 subjects. Each subject has 180 volumes of data during which they perform an N-back experiment as described in Section 4.1.

Description of the Experiment

First, let us load some packages that are required to run this analysis. The package nlme has functions that will allow us to fit mixed effects models.

```
library(nlme)
```

We will also load the package fmri which contains a function that allows us to convolve our stimulus, creating an EV.

```
library(fmri)
```

Let us specify some of the basic parameters of our experiment.

n is the number of subjects.

```
n <- 10
```

As above, T is the length of our voxel-wise time-series (i.e. number of volumes per subject). We will collect 180 volumes per subject. Let us assume that each volume is acquired over 2 seconds. This means that the duration in time of our subject scan is 6 minutes.

```
T <- 180
```

In 4.1, we proposed an example task with 4 conditions, and assumed one task run per subject. The conditions are as follows:

- EV_b (Fixation cross): Subject sees a white fixation cross on a black screen and lets thoughts wander.
- EV_1 (0-Back): Subject sees a series of letters, and presses a button when the letter is "X".
- EV_2 (1-Back): Subject sees a series of letters, and presses a button when the letter is the same as the one that preceded it. This task engages working memory.

- EV_3 (2-Back): Subject sees a series of letters, and presses a button when the letter is the same as the one that was presented two letters ago. This task engages working memory and is more difficult than 1-Back.

Let us assume that we present each subject first with the fixation cross, then with a 0-back condition, a 1-back condition, and then a 2-back condition for 15 volumes (30 seconds) for each condition. We will repeat this sequence (60 volumes total, or 120 seconds) three times. Normally, the order of the conditions would be randomized to prevent order-dependent effects, and we would need to obtain more data to achieve enough statistical power to actually detect an effect. However, the design presented here is easy to visualize, and so we will continue with these caveats.

It is now easy to describe our experiment stimuli with sequences of ones and zeros. The stimuli corresponding to each condition are set to one for each volume that the subject is presented with the condition, and zero at all other times.

```
emptyblock <- rep(0,15)
taskblock <- rep(1,15)
EVstim1 <- rep(c(emptyblock,taskblock,emptyblock,emptyblock),3)
EVstim2 <- rep(c(emptyblock,emptyblock,taskblock,emptyblock),3)
EVstim3 <- rep(c(emptyblock,emptyblock,emptyblock,taskblock),3)
```

We can plot these stimuli (Figure 2).

```
plot(1:T, EVstim1,type="n", ylab="", xlab="Volume Number")
lines(1:T,EVstim1)
lines(1:T,EVstim2,col="red")
lines(1:T,EVstim3,col="blue")
legend("topright", c("EVstim1", "EVstim2", "EVstim3"), lty=c(1,1,1), col=c("b
lack","red", "blue", cex=.75))
```

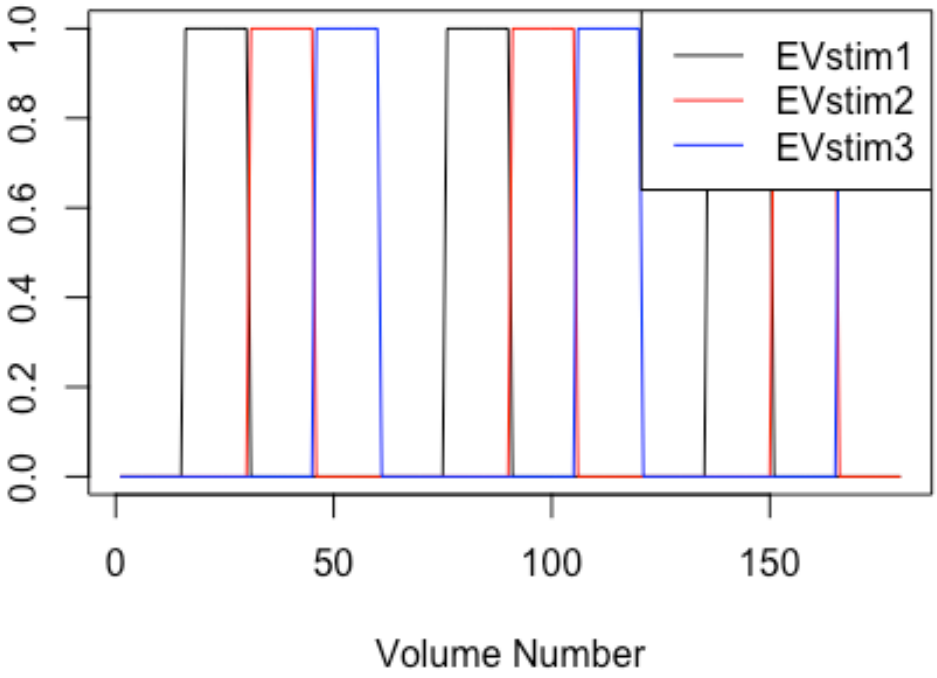


Figure 2 Boxcar plot for EV1, EV2 and EV3

While it is easy to see with this "boxcar" plot what the timings of the stimuli are, these stimuli need to be convolved with a hemodynamic response function to account for the delay between when stimuli are presented and when the BOLD response peaks. This can be done with the fmri.stimulus function from the fmri package. We use the canonical hemodynamic function as an example below.

```
baseonsets <- c(15, 15+60, 15+2*60)
EV1 <- fmri.stimulus(scans=T, onsets=baseonsets,durations=c(15,15,15))
EV2 <- fmri.stimulus(scans=T, onsets=(baseonsets+15),durations=c(15,15,15))
EV3 <- fmri.stimulus(scans=T, onsets=(baseonsets+30),durations=c(15,15,15))
```

We can plot these EVs to visualize them (Figure 3).

```
plot(1:T, EV1,type="n", ylab="", xlab="Volume Number")
lines(1:T,EV1)
lines(1:T,EV2,col="red")
lines(1:T,EV3,col="blue")
legend("topright", c("EV1", "EV2", "EV3"), lty=c(1,1,1), col=c("black","red",
"blue", cex=.75))
```

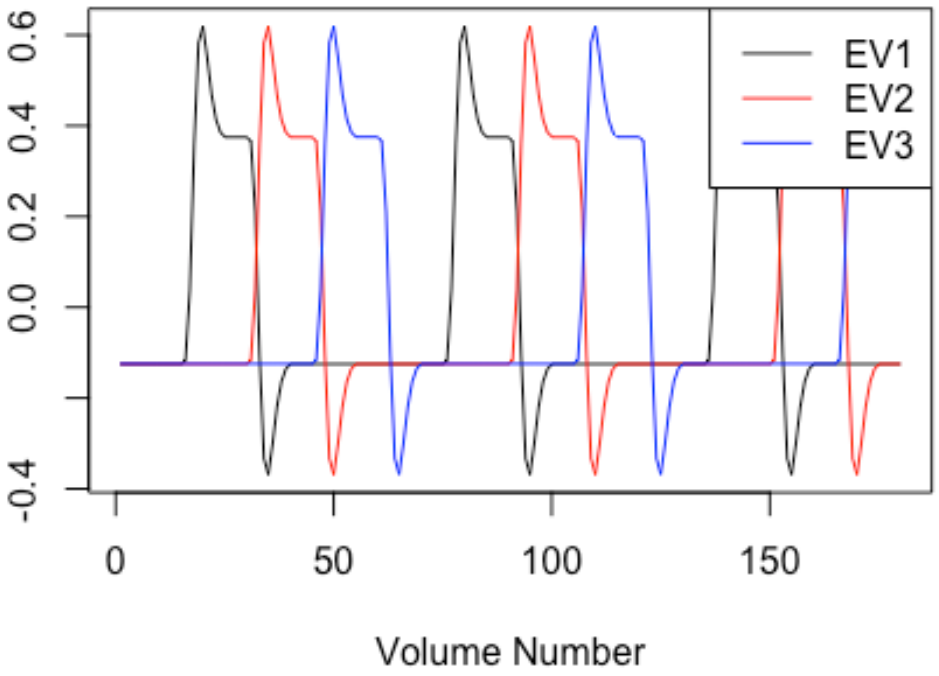


Figure 3 EVs convolved with the hemodynamic response function

In summary, we expect the BOLD signal to be related to these EVs in voxels that are associated with these task conditions.

Simulating Individual Subject Data

Now that we have created our design matrix, we need to simulate data for one voxel in 10 subjects. Recall the first level model for the BOLD signal at each voxel (1).

$$Y_k = X_k\beta_k + e_k = 1\beta_0 + X_{k1}\beta_1 + X_{k2}\beta_2 + X_{k3}\beta_3 + e_k \qquad (8)$$

For each subject k, where $k = 1 \ldots N$, the above equation models the BOLD signal at a single voxel, Y_k. X_k is our design matrix of EVs that we created in the description of the experiment (Section 4.1), β_k is a column vector of beta co-efficients (that the model tries to estimate) for each EV in the design matrix X_k, and e_k is a column vector of residuals for each voxel.

Remember that we do not model the baseline EV_b because doing so will produce a rank deficient design matrix (see The General Linear Model), but we do model EV_1, EV_2 and EV_3. Including the intercept, the design matrix for the first level (X_k) has four columns. We assume that there are T=180 volumes of fMRI data in the fMRI timeseries for each subject.

$$X_k = \begin{bmatrix} 1 & EV_{11} & EV_{21} & EV_{31} \\ 1 & EV_{12} & EV_{22} & EV_{32} \\ 1 & EV_{13} & EV_{23} & EV_{33} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & EV_{1T} & EV_{2T} & EV_{3T} \end{bmatrix} = [X_{k0} \quad X_{k1} \quad X_{k2} \quad X_{k3}] \qquad (9)$$

To simulate voxel-wise data using this matrix, we will specify fixed values for the coefficients for the intercept & EVs.

```
w
```

Thus, for subject 1,

$$\beta_1 = \begin{bmatrix} \beta_{10} \\ \beta_{11} \\ \beta_{12} \\ \beta_{13} \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 5 \\ 4 \end{bmatrix} \qquad (10)$$

The only thing we are missing now is e_k, the residual within-subject variance. Let us assume that this follows a normal distribution with a standard deviation for each subject of 0.5.

```
within.sd <- 0.5
```

We can now simulate the time series for a single voxel for the first subject.

```
y1 <- beta0.fixed + EV1*beta1.fixed + EV2*beta2.fixed + EV3*beta3.fixed + rno
rm(T, 0, within.sd)
```

Let's have a look at the voxel time series (y1) for the first 75 time points for subject 1.

```
##  [1] 2.0214509 1.3490615 1.8557519 1.2468316 1.2927957 1.8039662 0.9376718
##  [8] 1.3853797 1.7611807 1.4323804 1.4288028 1.4643618 1.2307101 1.5522157
## [15] 2.6619779 1.4431067 1.9772349 2.8259967 3.8067563 3.0598022 3.3062238
## [22] 2.6054416 2.6412604 3.0828330 2.2234654 3.0709759 2.9885094 2.4995354
## [29] 3.4957870 2.9433984 2.5864614 2.1906240 3.0916448 4.1660110 5.0909775
## [36] 5.3910551 4.1246516 4.8307661 4.3736932 3.6250929 4.2696208 3.6418103
## [43] 4.2807231 3.6383935 5.2752675 4.0340188 4.3373802 3.0997501 2.5263645
## [50] 3.4229418 3.1097041 4.1433298 3.6107748 4.0605794 3.5522102 3.6950509
## [57] 3.0570028 3.6373566 3.4647724 2.8855155 4.0103560 3.3063476 2.3284840
## [64] 1.5908046 1.1028553 0.9443586 2.0049827 2.1502510 1.6920226 0.6565557
## [71] 2.1916411 1.7191809 1.6878545 1.7996973 1.2873863
```

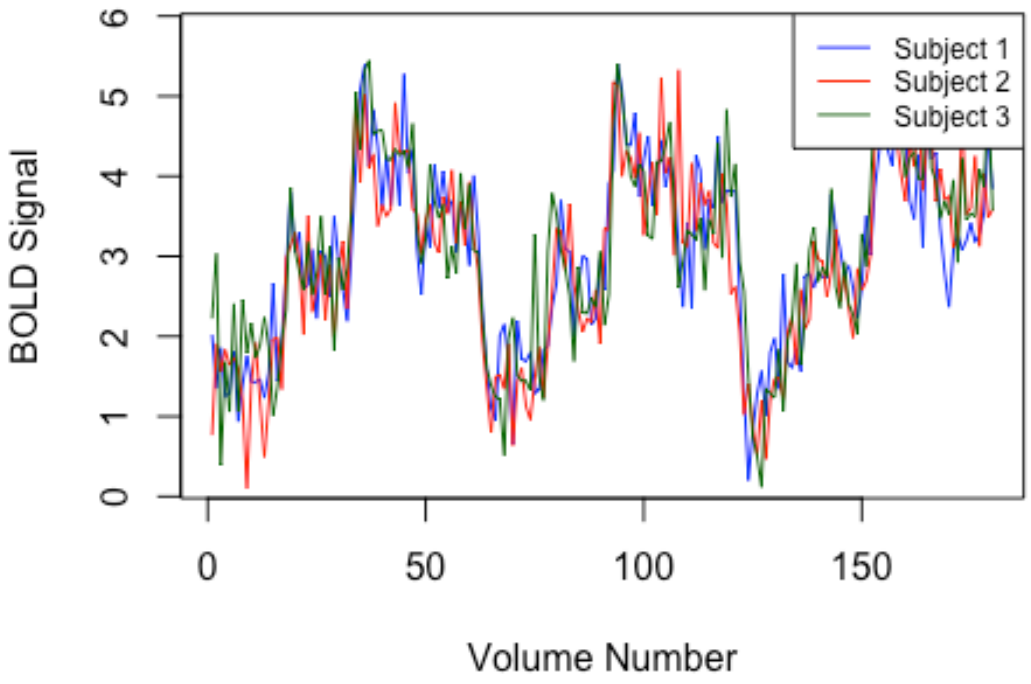
We can simulate this voxel time series in the same way for the second subject.

```
y2 <- beta0.fixed + EV1*beta1.fixed + EV2*beta2.fixed + EV3*beta3.fixed + rno
rm(T, 0, within.sd)
```

And for the third subject.

```
y3 <- beta0.fixed + EV1*beta1.fixed + EV2*beta2.fixed + EV3*beta3.fixed + rno
rm(T, 0, within.sd)
```

Now, we can look at the BOLD signal for the first 3 subjects for a single voxel (4).



Each of these subjects responds exactly the same way to the EVs, with the same degree of individual variation. In reality, this is unlikely to be the case. We need to simulate more realistic individuals by allowing their β values to vary around this mean value.

Group-level Analysis

Now we can simulate the data for all subjects, and add between-subject variation for the beta weights. Then, we will perform a group-level analysis for that single voxel so that we can generalize any hypothesis testing we do to the population of interest. Note that actual fMRI analysis is much more complex because we are usually modeling thousands of voxels at once, as opposed to a single voxel, for a larger subject pool.

A single voxel can be modeled according to the following equation at the group level (7).

$$Y_g = 1\beta_0 + X_1\beta_1 + X_2\beta_2 + X_3\beta_3 + b_0 + X_1(mat)b_1 + X_2(mat)b_2 + X_3(mat)b_3 + e \qquad (11)$$

Where 1 is a column vector of ones, $X_1 = (X_{11}, \dots X_{N1})'$, $X_2 = (X_{12}, \dots X_{N2})'$, $X_3 = (X_{13}, \dots X_{N3})'$, $b_0 = (b_{10}, \dots, b_{N0})'$, $b_1 = (b_{11}, \dots, b_{N1})'$, $b_2 = (b_{12}, \dots, b_{N2})'$, $b_3 = (b_{12}, \dots, b_{N3})'$, $X_1(mat) = diag(X_{11}, X_{21}, \dots, X_{N1})$, $X_2(mat) = diag(X_{12}, X_{22}, \dots, X_{N2})$, $X_3(mat) = diag(X_{13}, X_{23}, \dots, X_{N3})$. In other words, $X_1(mat)$, $X_2(mat)$ and $X_3(mat)$ are diagonal matrices with the column vectors being the diagonal entries.

In this model, $Y_g = (Y_1, Y_2, \ldots, Y_N)$ is a vector that contains the time series of the voxel for all subjects.

β_0 is the fixed intercept. β_1 represents the population effect for EV_1, β_2 is the population effect for EV_2, and β_3 is the population effect for $EV3$.

With mixed effects modeling, we can model the between-subjects variance around the population mean for each EV. These will be the random effects that we will later on specify in the model.

b_0 is the random effect for the intercept, b_{k1} is the random effect for EV_1 (i.e. it reflects subject to subject variability around the population effect β_1), b_{k2} is the random effect for EV_2, and b_{k3} is the random effect for EV_3. Finally, e_k represents within-subject error.

To simulate between-subjects variance, we assume a random intercept and slopes. We specify the variances of the intercept and slopes and then draw samples from their corresponding normal distributions. We set a seed so that this code will run the same way repeatedly.

```
set.seed(123)
sd0.random <- 1.5
sd1.random <- 1
sd2.random <- .5
sd3.random <- .8

beta0.betweensub <- rnorm(n, 0, sd0.random)
beta1.betweensub <- rnorm(n, 0, sd1.random)
beta2.betweensub <- rnorm(n, 0, sd2.random)
beta3.betweensub <- rnorm(n, 0, sd3.random)
```

The standard deviation of the intercept across individuals was set equal to sd0.random, or 1.5. Let's look at the actual random effect for our intercept, beta0.betweensub, for all 10 subjects.

```
##  [1] -0.8407135 -0.3452662  2.3380625  0.1057626  0.1939316  2.5725975
##  [7]  0.6913743 -1.8975919 -1.0302793 -0.6684930
```

The sample standard deviation of these values is 1.43.

Now, we will simulate the full timeseries Y_g at a single voxel. Recall that Y_g is the concatenated time series data for all of the N subjects.

To derive the voxel time series for each subject, we have to account for the fixed effects and random effects for each EV, the intercept, and within-subjects variability.

To create this vector, we define the function generateSubjectVoxelData, which creates the data for a single subject using the beta weights, as well as the between- and within-subjects error variances that we defined above. We then call this function for all n subjects using lapply. This returns the vectors for each subject as a list.

```
generateSubjectVoxelData <- function(i) {
  Y <- beta0.fixed + EV1*beta1.fixed + EV2*beta2.fixed + EV3*beta3.fixed + be
ta0.betweensub[i] + EV1*beta1.betweensub[i] + EV2*beta2.betweensub[i] + EV3*b
eta3.betweensub[i] + rnorm(T, 0, within.sd)
  return(Y)
}

allsubjs <- lapply(1:n, generateSubjectVoxelData)
```

The lme function that we will use later expects a data frame in the *long* data format. Figure 5 is a picture of what this looks like. This data frame will contain the voxel data Y.g, the EVs, and the identifier for all subjects "stacked" on top of each other, so that each row contains the value of the BOLD signal at a single voxel for one volume. Thus, the entire data frame will contain N*T, or 1800 observations (Figure 5).

idnum	Y_{gk}	EV1	EV2	EV3
1	0.31023810	-0.12496047	-0.12496047	-0.12455801
1	0.55363296	-0.12496047	-0.12496047	-0.12455801
1	0.02489342	-0.12496047	-0.12496047	-0.12455801
1	1.74206958	-0.12496047	-0.12496047	-0.12455801
1	1.26157259	-0.12496047	-0.12496047	-0.12455801
1	0.09603730	-0.12496047	-0.12496047	-0.12455801
1	0.45614918	-0.12496047	-0.12496047	-0.12455801
1	0.42426392	-0.12496047	-0.12496047	-0.12455801
1	1.04757415	-0.12496047	-0.12496047	-0.12455801
⋮	⋮	⋮	⋮	⋮
2	0.9928511	-0.12496047	-0.12496047	-0.12455801
2	1.5888308	-0.12496047	-0.12496047	-0.12455801
2	1.8347619	-0.12496047	-0.12496047	-0.12455801
2	1.6336320	-0.12496047	-0.12496047	-0.12455801
2	1.0980092	-0.12496047	-0.12496047	-0.12455801
2	1.3097128	-0.12496047	-0.12496047	-0.12455801
⋮	⋮	⋮	⋮	⋮

Figure 5 Long data frame with example voxel data for all subjects; idnum is the subject number.

Our first step towards creating this data frame is to create a single vector, Y.g, which is equivalent to Y_g from the equation above. This is a vector that contains the time series of the voxel across all subjects. To do this, we collapse the list of subject voxel vectors using unlist.

```
Y.g <- unlist(allsubjs)
```

We will not print the entire variable Y.g because it is a very long vector that will take up a lot of space. However, we can look at a particular subject's voxel time series by subsetting the vector. For example, subject 2's voxel time series is located in indices (T+1 to 2*T).

```
print(Y.g[(T+1):(2*T)])
```

```
##   [1]  0.9928511  1.5888308  1.8347619  1.6336320  1.0980092  1.3097128
##   [7]  0.9275396  0.9212288  1.7221631  0.7720416  2.2574848  1.2346781
##  [13]  1.3871073  0.9105740  0.9926435  0.6417074  1.5744744  2.6305198
##  [19]  3.1121797  2.6437986  2.4773796  2.4295903  3.3003317  1.9182851
##  [25]  2.3698458  3.4105526  2.4088842  1.7794513  2.1269869  2.7021016
##  [31]  2.2934278  2.5925737  3.5114844  4.2961096  5.1407948  4.1225524
##  [37]  4.5023620  4.1396268  3.6973180  2.9581198  3.4640121  3.4796355
##  [43]  3.7481480  4.3746701  4.8711103  4.4886287  3.4735906  2.3012634
##  [49]  2.7324822  2.9310990  3.3867931  3.5367745  3.4589847  2.4504115
##  [55]  3.5860356  2.9379355  3.2486155  3.1984897  3.3752975  3.1735516
##  [61]  2.2949739  2.9136274  1.5340090  0.3644077  0.4214640  0.6892015
##  [67]  1.0373417  1.9520364  1.1263457  1.3638705  1.8640298  1.8069284
##  [73]  1.8524694  0.9911039  2.2810792  1.3335659  2.5993630  1.7455772
##  [79]  2.9605193  3.6595242  2.5140695  2.3043451  2.0830321  1.9606802
##  [85]  2.2407919  2.6249612  1.4522664  2.5653619  3.0777092  3.4781586
##  [91]  3.1318172  3.2518992  2.8200778  3.9501079  4.1655172  4.5185969
##  [97]  3.9091266  3.1949254  4.5963558  4.1802664  3.8432859  4.3336251
## [103]  3.0551836  4.0549809  3.4631146  4.0567110  3.5097551  3.4960075
## [109]  3.5946310  2.8901404  3.4730659  2.4612935  2.7560277  3.9076576
## [115]  3.3499081  2.1351027  2.4791954  3.0608236  3.5941038  3.1102725
## [121]  3.4408052  3.0248821  2.1765231  0.5252418  0.3364008 -0.2544365
## [127]  0.9764957  0.8456883  0.7488661  1.1898847  1.7873094  0.2834636
## [133]  1.0661982  1.3381565  0.8332341  1.4671670  1.8716520  2.4045105
## [139]  1.7170784  4.3202960  2.7690410  3.0062863  2.6891847  2.9992735
## [145]  2.8682014  2.3544751  2.6484555  1.9866672  2.8878331  2.2288525
## [151]  3.6896159  2.0479874  3.4514494  4.6635512  4.5966067  3.8720945
## [157]  3.4635719  3.8964876  3.7469843  2.8294302  3.7418463  3.8196859
## [163]  3.8119340  3.1970623  3.9626374  4.4041233  3.7682843  2.6117329
## [169]  2.7090494  3.0595472  2.6407638  1.9766874  3.5589554  2.7333099
## [175]  2.8744340  3.9131644  2.7741417  3.5840799  2.5308727  2.9839429
```

Our subjects are numbered 1 to 10. We need to generate a list of subject identifiers to correspond to the voxel data in Y.g. By convention, the vector of subject identifiers is called idnum. We convert these integer subject identifiers to factors.

```
idnum <- unlist(lapply(1:n,rep,T))
idnum <- as.factor(idnum)
```

Similarly, we need to replicate the EV vectors by the number of subjects to create this data frame.

```
EV1.vec <- rep(EV1, n)
EV2.vec <- rep(EV2, n)
EV3.vec <- rep(EV3, n)
```

We can create our data frame voxeldat, for convenience. If we had additional nuisance covariates, they would be additional columns in this structure.

```
voxeldat <- data.frame(idnum, Y.g, EV1.vec, EV2.vec, EV3.vec)
```

Now, let's use the lme function from the nlme package to fit the group level model for our single voxel.

```
model1 <- lme(Y.g ~ 1 + EV1.vec + EV2.vec + EV3.vec,
              random = ~1 + EV1.vec + EV2.vec + EV3.vec  | idnum, data=voxeld
at, method=c("ML"))

summary(model1)
```

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: voxeldat
##       AIC      BIC   logLik
##   2763.74 2846.173 -1366.87
##
## Random effects:
##  Formula: ~1 + EV1.vec + EV2.vec + EV3.vec | idnum
##  Structure: General positive-definite, Log-Cholesky parametrization
##             StdDev    Corr
## (Intercept) 1.3604635 (Intr) EV1.vc EV2.vc
## EV1.vec     0.9674509  0.635
## EV2.vec     0.4333729 -0.516 -0.900
## EV3.vec     0.4048077  0.648  0.280 -0.452
## Residual    0.4987049
##
## Fixed effects: Y.g ~ 1 + EV1.vec + EV2.vec + EV3.vec
##                Value Std.Error   DF   t-value p-value
## (Intercept) 3.123745 0.4308559 1787  7.250092       0
## EV1.vec     2.241342 0.3116205 1787  7.192537       0
## EV2.vec     4.709825 0.1497918 1787 31.442480       0
## EV3.vec     4.239160 0.1410563 1787 30.052968       0
##  Correlation:
##         (Intr) EV1.vc EV2.vc
## EV1.vec  0.624
## EV2.vec -0.472 -0.771
## EV3.vec  0.588  0.286 -0.286
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -2.906193178 -0.679614803  0.001845878  0.666592422  3.394353801
```

```
##
## Number of Observations: 1800
## Number of Groups: 10
```

We simulated our data without autocorrelation. In real fMRI data, autocorrelation is typically present. One way to account for autocorrelation in our linear mixed effects analysis is to use the argument correlation to specify the assumed correlation structure (Pinheiro et al. 2017). As an example, for an autoregressive process of order one, we could specify its form in the call to lme as follows.

```
model2 <- lme(Y.g ~ 1 + EV1.vec + EV2.vec + EV3.vec,
              random = ~1 + EV1.vec + EV2.vec + EV3.vec  | idnum, data=voxeld
at, method=c("ML"), correlation=corAR1(form=~1|idnum))
```

We can compare these two models, and as expected, find that by the AIC, the BIC, and the log-likelihood ratio, model 1 (without autocorrelation) is superior to model 2 (with autocorrelation).

```
anova(model1,model2)
```

```
##        Model df      AIC      BIC    logLik   Test   L.Ratio p-value
## model1     1 15 2763.740 2846.173 -1366.870
## model2     2 16 2765.384 2853.313 -1366.692 1 vs 2 0.3559728  0.5508
```

We will focus now only on model 1. When we look at the fitted beta coefficients for the intercept and the EVs, we find that they are pretty close to the mean of the true beta values that we had specified above. To refresh your memory, our fixed beta values were β_0=3, β_1=2, β_2=5, and β_3=4.

The estimated fixed effects match the specified values closely. They are unlikely to match perfectly, unless the sample size is every large.

```
## (Intercept)     EV1.vec     EV2.vec     EV3.vec
##       3.124       2.241       4.710       4.239
```

We can also compare the estimated standard deviations of the model to the true standard deviations that we had specified. To do this, we look at the printout summary of the model.

Under the *Random effects* section of the output from lme, the first column gives a readout of the estimated standard deviations. We can also see estimated standard deviations using the VarCorr function.

```
VarCorr(model1)
```

```
## idnum = pdLogChol(1 + EV1.vec + EV2.vec + EV3.vec)
##             Variance  StdDev    Corr
## (Intercept) 1.8508608 1.3604635 (Intr) EV1.vc EV2.vc
## EV1.vec     0.9359612 0.9674509  0.635
## EV2.vec     0.1878120 0.4333729 -0.516 -0.900
## EV3.vec     0.1638693 0.4048077  0.648  0.280 -0.452
## Residual    0.2487066 0.4987049
```

As a reminder, let us look at the standard deviations of the simulated data.

```
sd(beta0.betweensub)

## [1] 1.430676

sd(beta1.betweensub)

## [1] 1.038073

sd(beta2.betweensub)

## [1] 0.4654046

sd(beta3.betweensub)

## [1] 0.4218419

within.sd

## [1] 0.5
```

We see that the standard deviations estimated for the random effect match closely with the corresponding standard deviations of the simulated data. The standard deviations of the random effects also agree well with the true values. (It is not expected that the estimates will agree exactly with the true values.) For reference, these were:

```
sd0.random

## [1] 1.5

sd1.random

## [1] 1

sd2.random

## [1] 0.5

sd3.random

## [1] 0.8
```

Testing a Hypothesis (for a Specific Contrast)

Let us assume that we want to test whether the 1-Back condition is associated with greater activation in the population than the 0-Back condition. Statistically, this means we need to test whether EV_2 is greater than EV_1 at the population level. Our null hypothesis would be that there is no difference in the BOLD signal of the voxel between the two conditions (i.e., that they are equal). Recall that in our simulations, we set $\beta_1 = 2$ and $\beta_2 = 5$.

Therefore,

$H_0: EV_2 = EV_1$

$H_1: EV_2 > EV_1$

To ask whether $EV_2 > EV_1$, we could reformulate the model so that the effect of this contrast is estimated directly. This is done by simple algebraic manipulation. In our first-level model, we could write

$$\begin{aligned} Y &= \beta_0 + X_1\beta_1 + X_2\beta_2 + X_3\beta_3 + e \\ &= \beta_0 + (X_1 + X_2 - X_2)\beta_1 + X_2\beta_2 + X_3\beta_3 + e \\ &= \beta_0 + X_1\beta_1 + X_2\beta_1 - X_2\beta_1 + X_2\beta_2 + X_3\beta_3 + e \\ &= \beta_0 + (X_1 + X_2)\beta_1 + X_2(\beta_2 - \beta_1) + X_3\beta_3 + e \end{aligned} \quad (12)$$

Basically, we have now changed the EVs so that the β weight for the second EV now directly tests the contrast of interest. We can interpret the significance of this coefficient directly.

In R, this adjusted model specification is as follows (note that we do not make use of the voxeldat data structure here and just refer to the vectors that we create).

```
EV1.adj <- EV1.vec + EV2.vec
EV2.adj <- EV2.vec
model3 <- lme(Y.g ~ 1 + EV1.adj + EV2.adj + EV3.vec, random = ~1 + EV1.adj +
EV2.adj + EV3.vec|idnum, method=c("ML"))
```

Let's have a look at the results.

```
## Linear mixed-effects model fit by maximum likelihood
##  Data: NULL
##       AIC      BIC   logLik
##   2763.74 2846.173 -1366.87
##
## Random effects:
##  Formula: ~1 + EV1.adj + EV2.adj + EV3.vec | idnum
##  Structure: General positive-definite, Log-Cholesky parametrization
##             StdDev    Corr
## (Intercept) 1.3605036 (Intr) EV1.dj EV2.dj
## EV1.adj     0.9674567  0.635
## EV2.adj     1.3707182 -0.612 -0.991
## EV3.vec     0.4048042  0.648  0.280 -0.340
## Residual    0.4987050
##
## Fixed effects: Y.g ~ 1 + EV1.adj + EV2.adj + EV3.vec
##                Value Std.Error   DF   t-value p-value
## (Intercept) 3.123745 0.4308686 1787  7.249879       0
## EV1.adj     2.241342 0.3116223 1787  7.192496       0
## EV2.adj     2.468484 0.4376875 1787  5.639832       0
## EV3.vec     4.239160 0.1410553 1787 30.053186       0
##  Correlation:
##         (Intr) EV1.dj EV2.dj
## EV1.adj  0.624
## EV2.adj -0.606 -0.976
```

```
## EV3.vec  0.588  0.286 -0.301
##
## Standardized Within-Group Residuals:
##          Min           Q1          Med           Q3          Max
## -2.906188544 -0.679620040  0.001849009  0.666592527  3.394350168
##
## Number of Observations: 1800
## Number of Groups: 10
```

The t-statistic and the p-value for this contrast are now the t-statistic and p-value for the fixed effect for EV2.adj. We see here that our p-value is significant if we take α=0.05. In other words, we can reject the null hypothesis.

You could write your EVs from the beginning of your experimental design to conform to the contrasts of interest in the way shown above, but this kind of model rewriting of the model can be difficult and error prone. It is normally more straightforward to specify the EVs in a logical way and to specify the contrast directly.

In the following example, we use model 1 and set contr to be the linear combination EV_2 - EV_1. Then, the t-statistic is calculated by dividing the linear combination of the parameter estimates (as specified by the contrast) by the square root of the variance of this linear combination.

```
contr <- c(0, -1, 1, 0)
out <- anova(model1, L=contr)
tstat <- t(contr) %*% model1$coefficients$fixed / sqrt(t(contr) %*% vcov(mode
l1) %*% contr)
pvalue <- 1 - pt(tstat, df=out$denDF)
```

We can see that the values obtained by specifying the contrast directly are indeed the same as we had obtained by recoding the contrast (tstat = 5.65 and pvalue = 0).

Conclusions

We have described the structure of fMRI data, issues in pre-processing, the mixed effects modeling framework used to analyze fMRI data, and how it is implemented in a common specialized fMRI analysis package (FSL) and in R. Further, we have provided a self-contained tutorial that both simulates data for a single voxel and demonstrates all the steps necessary to create a design matrix, model the data, and evaluate a contrast using R. Thus in a single document we have presented the information necessary to translate the language of specialized fMRI software and analysis into an R framework. Our hope is that this tutorial can foster interdisciplinary communication through the common and open R statistical platform. Of note, the tutorial itself is written as a self-contained *Rmarkdown* document that can be used as an example for reproducible analyses (see Supplemental Materials).

To emphasize the most important points, we have left out others. Part of the difficulty in communicating the complexity of fMRI analyses to those in other disciplines stems from

subtle aspects of statistical methodology and an incomplete understanding of how variations in fMRI pre-processing can alter assumptions made downstream. This document does not address these issues; for some excellent discussion of that topic, see (Carp 2012; Churchill et al. 2017). Importantly, because we focus on modeling a single voxel, this tutorial does not cover the vast literature on correction for multiple comparisons, which is complicated and nuanced (Chen, Taylor, and Cox 2017; Eklund, Nichols, and Knutsson 2016). We also do not describe common neuroimaging file formats or how to read them into R, but this is readily accomplished by established R packages such as oro.nifti (Whitcher, Schmid, and Thornton 2011). Finally, the same model is likely to take far longer to run in R than in specialized software packages. At a minimum, parallelization is required to reduce the overall execution time. This can be accomplished in R but is not described here.

References

Askren, Mary K., Trevor K. McAllister-Day, Natalie Koh, Zoé Mestre, Jennifer N. Dines, Benjamin A. Korman, Susan J. Melhorn, et al. 2016. "Using Make for Reproducible and Parallel Neuroimaging Workflow and Quality-Assurance." *Frontiers in Neuroinformatics* 10: 2. doi:10.3389/fninf.2016.00002.

Bright, Molly G., and Kevin Murphy. 2015. "Is fMRI 'Noise' Really Noise? Resting State Nuisance Regressors Remove Variance with Network Structure." *NeuroImage* 114 (July): 158–69. doi:10.1016/j.neuroimage.2015.03.070.

Carp, Joshua. 2012. "On the Plurality of (Methodological) Worlds: Estimating the Analytic Flexibility of FMRI Experiments." *Frontiers in Neuroscience* 6: 149. doi:10.3389/fnins.2012.00149.

Chen, Gang, Paul A. Taylor, and Robert W. Cox. 2017. "Is the Statistic Value All We Should Care About in Neuroimaging?" *NeuroImage* 147 (February): 952–59. doi:10.1016/j.neuroimage.2016.09.066.

Cheong, JeeWon, David P. MacKinnon, and Siek Toon Khoo. 2003. "Investigation of Mediational Processes Using Parallel Process Latent Growth Curve Modeling." *Structural Equation Modeling : A Multidisciplinary Journal* 10 (2): 238. doi:10.1207/S15328007SEM1002_5.

Churchill, Nathan W., Pradeep Raamana, Robyn Spring, and Stephen C. Strother. 2017. "Optimizing fMRI Preprocessing Pipelines for Block-Design Tasks as a Function of Age." *NeuroImage* 154 (July): 240–54. doi:10.1016/j.neuroimage.2017.02.028.

Cox, Robert W. 1996. "AFNI: Software for Analysis and Visualization of Functional Magnetic Resonance Neuroimages." *Computers and Biomedical Research* 29 (3). Elsevier: 162–73.

Eklund, Anders, Thomas Nichols, and Hans Knutsson. 2016. "Cluster Failure: Why fMRI Inferences for Spatial Extent Have Inflated False Positive Rates." *Proceedings of the National Academy of Sciences* 113 (28): 7900–7905.

Friston, Karl, John Ashburner, Stefan Kiebel, Thomas Nichols, and William Penny, eds. 2007. *Statistical Parametric Mapping: The Analysis of Functional Brain Images*. 1st ed. Elsevier.

Gandrud, Christopher. 2016. *Reproducible Research with R and R Studio, Second Edition*. CRC Press.

Gentleman, Robert C, Vincent J Carey, Douglas M Bates, Ben Bolstad, Marcel Dettling, Sandrine Dudoit, Byron Ellis, et al. 2004. "Bioconductor: Open Software Development for Computational Biology and Bioinformatics." *Genome Biology* 5 (10): R80. doi:10.1186/gb-2004-5-10-r80.

Gorgolewski, Krzysztof, Christopher D. Burns, Cindee Madison, Dav Clark, Yaroslav O. Halchenko, Michael L. Waskom, and Satrajit S. Ghosh. 2011. "Nipype: A Flexible, Lightweight and Extensible Neuroimaging Data Processing Framework in Python." *Frontiers in Neuroinformatics* 5. doi:10.3389/fninf.2011.00013.

Huettel, Scott A., Allen W. Song, and Gregory McCarthy. 2014. *Functional Magnetic Resonance Imaging, Third Edition*. 3rd edition. Sunderland, Massachusetts, U.S.A: Sinauer Associates, Inc.

Jenkinson, Mark, C Beckmann, Timothy EJ Beherens, M. Woolrich, and Stephen M Smith. 2012. "Fsl." *Neuroimage* 62 (2). Elsevier: 782–90.

Kamvar, Zhian N., Margarita M. López-Uribe, Simone Coughlan, Niklaus J. Grünwald, Hilmar Lapp, and Stéphanie Manel. 2017. "Developing Educational Resources for Population Genetics in R: An Open and Collaborative Approach." *Molecular Ecology Resources* 17 (1): 120–28. doi:10.1111/1755-0998.12558.

Laird, N. M., and J. H. Ware. 1982. "Random-Effects Models for Longitudinal Data." *Biometrics* 38 (4): 963–74.

Lindquist, Martin A. 2008. "The Statistical Analysis of fMRI Data." *Statist. Sci.* 23 (4). The Institute of Mathematical Statistics: 439–64. doi:10.1214/09-STS282.

Madhyastha, Tara, Matthew Peverill, Natalie Koh, Connor McCabe, John Flournoy, Kate Mills, and Kevin King. –. "Current Methods and Limitations for Longitudinal fMRI Analysis Across Development." *Developmental Cognitive Neuroscience*.

Mizumoto, Atsushi, and Luke Plonsky. 2016. "R as a Lingua Franca: Advantages of Using R for Quantitative Research in Applied Linguistics." *Applied Linguistics* 37 (2): 284–91. doi:10.1093/applin/amv025.

Monti, Martin M. 2011. "Statistical Analysis of fMRI Time-Series: A Critical Review of the GLM Approach." *Frontiers in Human Neuroscience* 5 (March). doi:10.3389/fnhum.2011.00028.

Newsom, Jason T. 2015. *Longitudinal Structural Equation Modeling: A Comprehensive Introduction*. New York: Routledge.

Open Science Collaboration. 2015. “Estimating the Reproducibility of Psychological Science.” *Science* 349 (6251). American Association for the Advancement of Science. doi:10.1126/science.aac4716.

Pernet, Cyril R. 2014. “Misconceptions in the Use of the General Linear Model Applied to Functional MRI: A Tutorial for Junior Neuro-Imagers.” *Frontiers in Neuroscience* 8. doi:10.3389/fnins.2014.00001.

Pinheiro, Jose, Douglas Bates, Saikat DebRoy, Deepayan Sarkar, and R Core Team. 2017. *Nlme: Linear and Nonlinear Mixed Effects Models*. https://CRAN.R-project.org/package=nlme.

Poldrack, Russell A., Jeanette A. Mumford, and Thomas E. Nichols. 2011. *Handbook of Functional MRI Data Analysis*. 1 edition. New York: Cambridge University Press.

Poline, Jean-Baptiste, and Matthew Brett. 2012. “The General Linear Model and fMRI: Does Love Last Forever?” *NeuroImage* 62 (2): 871–80. doi:10.1016/j.neuroimage.2012.01.133.

R Core Team. 2014. “R: A Language and Environment for Statistical Computing.” Vienna, Austria: R Foundation for Statistical Computing.

Whitcher, Brandon, Volker J. Schmid, and Andrew Thornton. 2011. “Working with the Dicom and Nifti Data Standards in R.” *Journal of Statistical Software*, no. 6.

Woolrich, M., Timothy E. J. Behrens, Christian F. Beckmann, Mark Jenkinson, and Stephen M. Smith. 2004. “Multilevel Linear Modelling for FMRI Group Analysis Using Bayesian Inference.” *NeuroImage* 21 (4): 1732–47. doi:10.1016/j.neuroimage.2003.12.023.

Woolrich, M., B. D. Ripley, M. Brady, and S. M. Smith. 2001. “Temporal Autocorrelation in Univariate Linear Modeling of FMRI Data.” *NeuroImage* 14 (6): 1370–86. doi:10.1006/nimg.2001.0931.