Dynamic Causal Modeling

Prepared by Ian Ballard

Motivated Memory Laboratory

Duke University

May, 2012

Contact: iancballard@gmail.com

Table of Contents

* Introduction to DCM
  + Disclaimer
  + Relevant Reading
  + Introduction to DCM (Theoretical)
  + Constructing a DCM (Practical)
  + The Bayesian Part (Theoretical)
  + Bayes Factor (Theoretical)
  + Model Comparison (Theoretical)
  + Advances (Practical)
* General overview of analysis stream (Practical)
* Other considerations (Practical)
  + Slice timing correction
  + Multiple Runs
  + Motion Effects
  + Normalization in FSL
  + Ordering of preprocessing
  + Directory structure
* Starting an analysis: Preprocessing: SPM and FSL
  + Brain extraction and intensity normalization
  + Single Subject: Realign and Smooth
  + Submitting Matlab Jobs to Xgrid
  + Group Data: Realign, Normalize, Smooth
* GLM within subject analysis in SPM
  + Defining onsets
  + Setting up and running GLMs
* GLM group level analysis in SPM
* Defining ROIs
  + Back-transform ROIs
  + Extract the principle eigenvariate
* Generate and Estimate DCMs
  + Define onset files
  + Set up a GLM
  + Construct DCM
  + Run DCMs
* Model Comparison
* Reporting Results
* Troubleshooting
  + Matlab Xgrid

Disclaimer

I am not a statistician! I haven’t taken a Bayesian statistics class. I am sure that the DCM experts would balk at some of the things in this introduction. I am sure that the concepts are right, and I try to explain them in laymen’s terms that make sense for me. If you are at the point of publishing, make sure to read some DCM papers and harvest the appropriate lingo.

Relevant Reading

Essential:

**10 simple rules for DCM**. PMID: 19914382.

*This gives a practical overview on how to use DCM*

Strongly Recommended:

**Critical Comments on Dynamic Causal Modeling** PMID: 22001162

**Model Selection and Gobbledygook** PMID: 22155029

*This is a critique of DCM and Friston’s response. The response explains some concepts in DCM really clearly, and the critique is interesting to keep in mind, because Friston responds fully to some, but not all, of their points.*

Recommended:

**Dynamic Causal Modeling**. PMID: 12948688

*This is the seminal paper introducing DCM.*

**Dynamic causal modelling: a critical review of the biophysical and statistical foundations**. PMID: 19961941.

*This gives a nice review of the updates and advances (nonlinear, stochastic, 2 state, et cetera)*

**SPM Manual**

*This goes through a DCM analysis, but it is outdated.*

Other (methods):

**Comparing families of dynamic causal models**. PMID: 20300649

*This explains the family level model comparison*

**Bayesian Model Selection for Group Studies**. PMID19306932

*This explains a new method for model comparison that employs random effects assumptions. This is a precursor to the above paper.*

**Network Discovery with DCM**. PMID: 21182971

*This explains how DCM could be used as an exploratory method for resting state data. As of now, this is not yet implemented in SPM.*

**Nonlinear Dynamic Causal Models for fMRI**. PMID 18565765

*This explains the nonlinear DCMs (where a region can influence the connection between 2 other regions during certain experimental contexts).*

**Generalized Filtering and Stochastic DCM for fMRI**. PMID: 21310247

*This advances DCM to include a model for the noise in the signal.*

**Dynamic and stochastic models of neuroimaging data: A comment on Lohmann et al.** PMID**:** 22387473

*This shows that stochastic models address the poor model fit shown by Lohmann et. al.*

Other (DCM versus granger):

**Causal modelling and brain connectivity in functional magnetic resonance imaging**. PMID: 19226186

*A paper by Friston comparing DCM to Granger causality.*

**The identification of interacting networks in the brain using fMRI: Model selection, causality and deconvolution.** PMID: 19786106

*A response to the above paper defending granger causality.*

**Dynamic causal modeling and Granger causality Comments on: The identification of interacting networks in the brain using fMRI: Model selection, causality and deconvolution**. PMID: 19770049

*A response to the response, apologizing for his “superficial” treatment of Granger causality in the first paper and responding to the critiques on DCM.*

Introduction to DCM

Dynamic causal modeling is an approach that lets you give alternative and (more) detailed accounts of activations observed in the GLM framework. The primary advantage is that you account for the fact that not only do experimental interactions influence a region, but that regions connections to other regions, as well as the functional sensitivity of those connections. A few examples of types of questions that could be answered by DCM include: what does this manipulation do to the connectivity between these regions?, are these regions connected in a directed or reciprocal fashion?, how does the intrinsic architecture of the network change in the presence of experimental manipulations, do they become less reciprocal and more directed?, where does information about this experimental manipulation enter the network?, how well does a class of models sharing an important feature fit Group 1’s data versus Group 2’s data?

DCM is causal in the sense that activity in region A impacts activity in region B; some portion of the activation in B would not occur if A were not to exist. It is also causal in the sense of timing; however, this is a bit fuzzier. First, different regions can have different regions, so A can happen before B in the neuronal state equation but the response to B can happen before the response to A in the predicted signal. More generally, DCM is just fitting data with a model that have temporal dynamics with a model that also has temporal dynamics. The models only give indirect insight into the temporal nature of the network and cannot in general tell you “what happens first.”

DCM is in a general sense not very different from general linear modeling (GLM). Both methods attempt to model brain activity as a function of experimental manipulations and known nuisance variables, fit these models to the data, and make inference on the fitted model parameters. They differ in the assumptions about how experimental effects manipulations rise to BOLD responses and how regions interact with one another.

In the case of a GLM, you attempt to model the observed BOLD timeseries of a voxel y as:

where X is a design matrix representing the effects you think contribute linearly to the bold response, B is a vector of beta weights, and e is an error term. A separate B is estimated for each voxel z, and you perform statistical tests on the estimated values of B in voxels of interest.

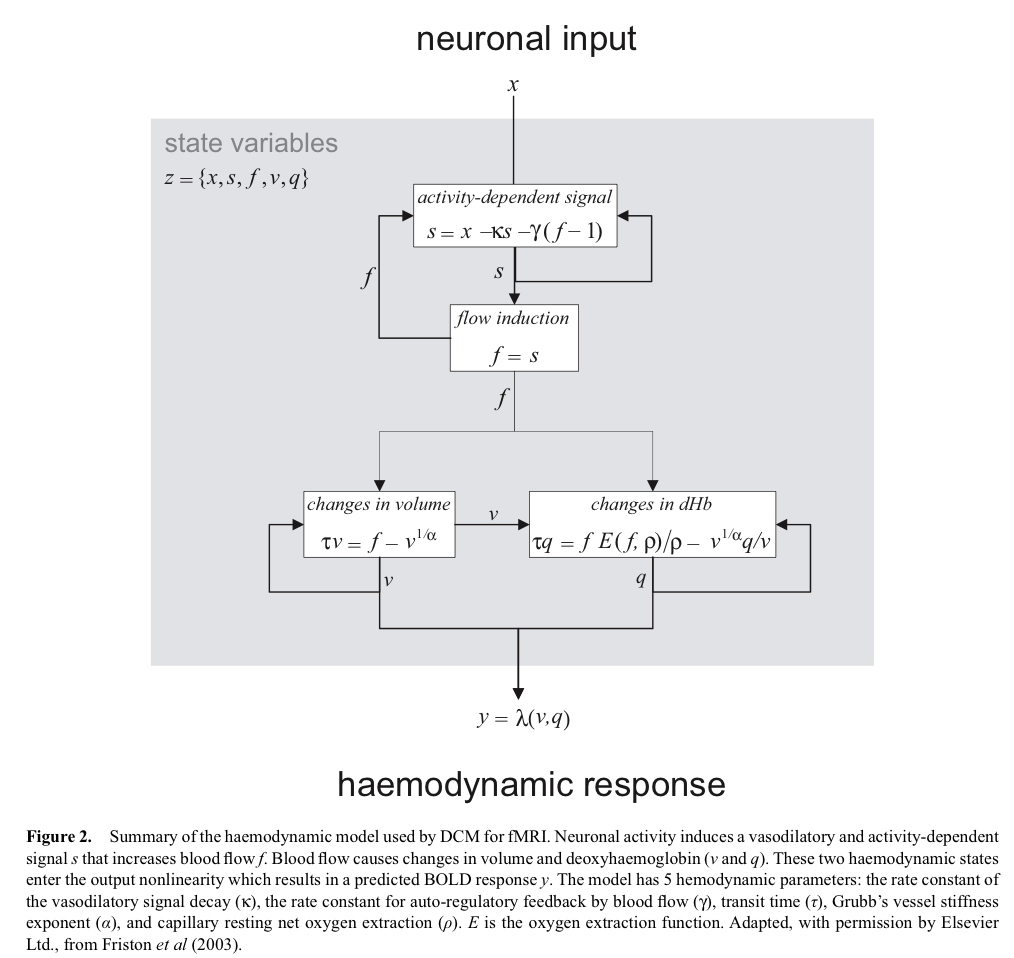
For dynamic causal modeling, one is still attempting to model the observed BOLD response in terms of experimental manipulations (often models the mean or first eigenvariate of several voxels in a region of interest (ROI) z, but one can model a single voxel as well):

Activity in a region is governed by three separate mechanisms: experimental events directly eliciting activation in a region (driving input: matrix C), connectivity from other regions (intrinsic connections: matrix A), and connectivity from other regions that is only present during specific experimental conditions (context sensitive connections, or modulatory connections: matrix B). We estimate a separate matrix B for each “modulatory” situation. For example, if you had a task where both faces and house could modulate a connection, you would have separate B matrices for faces and houses. A simply sets which regions are allowed to be connected. Note that for there to be a modulatory connection (B) between two regions there must also between an intrinsic connection (A) between these regions. However, the reverse is not true. u represents the experimental input, and j represents the different inputs that can have modulatory effects (you can think about as the different columns of a design matrix with j columns, before they are convolved with the HRF (eg. house and face regressors). It is interesting to note that if you set the A and B matrices go to zero, and adjust the convolution model, you recover the standard GLM model as a special case of DCM.

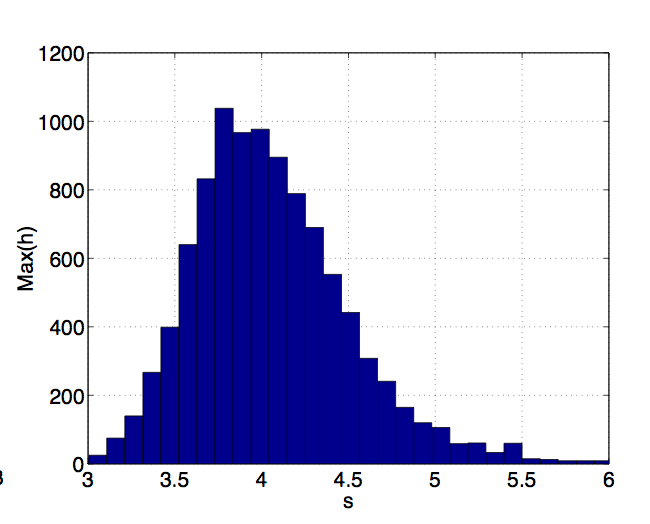
Also note that the equation specifies , which is the rate of change of activity in a region. Thus parameters have units of Hz, and a larger connection parameter corresponds to a faster influence of one region on another. The following steps are used to model the BOLD data:

1. Specify how experimental effects and connectivity influence
2. Integrate this equation to solve for z. z is then a predicted timecourse of the neuronal state across the experiment.
3. To model BOLD activity, these neuronal states are passed through a hemodynamic model (the balloon model), which models how neuronal input transforms into a BOLD response. (I’ve included a schematic of the balloon model below, but the details are, in my experience, unimportant)
4. This signal is then augmented with an error term and nuisance effects (ie. scanner drift), and together these are used to predict the observed data.

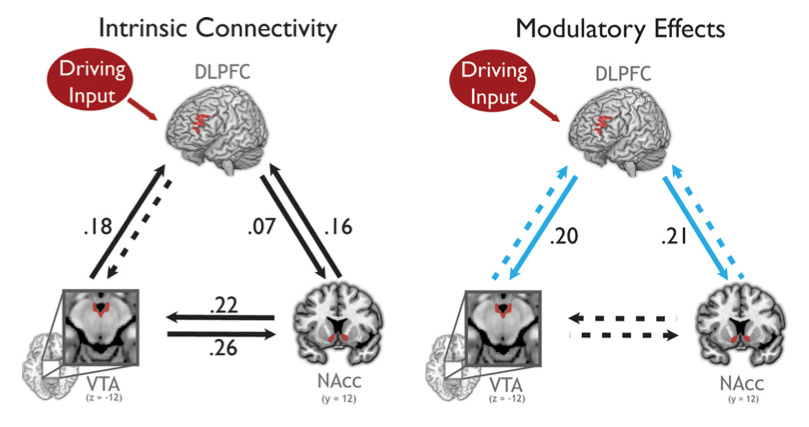
These steps provide a useful heuristic for thinking about DCM. However, it should be emphasized that steps 2-4 area really adjusted simultaneously during model estimation. In particular, hemodynamic parameters are optimized at the same time as the coupling (A,B, and C) parameters. This is important for understanding one major critique of DCM: the coupling parameters are to some degree dependent on the hemodynamic ones. You could imagine a scenario in which some hemodynamic parameter that you don’t care about could take on 2 different values for which (with different values of A,B, and C matrices) the equations provided an identical fit to the data. Thus, which A,B, and C measures you observe could depend on how model estimation decides to optimize hemodynamic parameters (this is also true for all other parameters that are to some degree correlated). The parameter values of the driving input (C matrix) are by far the most susceptible to this type of problem, and therefore you should avoid or be very cautious interpreting the strength of the driving input. Importantly, this is a generic problem for modeling, and not specific to DCM. Further, simulation studies have shown that DCM is relatively resilient to problems of this sort (e.g. comparing hemodynamic models with dcm).



One aspect of the balloon model that is important to understand is that it is estimated separately for each subject and each region. Thus, DCM can account for differential hemodynamic lags between regions in the brain. Below is a histogram out of 10,000 samples of the time to peak of the hemodynamic response according to the balloon model.



To return to the equations, we will illustrate how they breakdown outside of matrix notation using the “winning” model from the Ballard, Murty et. al paper. This model seemed to give the best account of our data. However, it is important to note that we estimated 112 models per subject, and we could equally look at any of those 112 models. This one has figures pre-made, so we will look at this one. We will turn to the issue of how to decide which models to run later on.



First we will set up the model, and then we will examine the estimated parameters. The model had full intrinsic connectivity (every region was allowed to be connected to every other region in both directions), the driving input of “all rewards” entered at the DLPFC and “high reward” events influenced the connections from DLPFC to VTA and to NAcc. Then the intrinsic connectivity matrix (The matrices are read: from column to row):

|  |  |  |  |
| --- | --- | --- | --- |
| Matrix A | VTA | NAcc | dlPFC |
| VTA | 1 | 1 | 1 |
| NAcc | 1 | 1 | 1 |
| dlPFC | 1 | 1 | 1 |

The modulatory matrix B allowed context-dependent connections from DLPFC to NAcc and from DLPFC to VTA:

|  |  |  |  |
| --- | --- | --- | --- |
| Matrix B | VTA | NAcc | dlPFC |
| VTA | 0 | 0 | 1 |
| NAcc | 0 | 0 | 1 |
| dlPFC | 0 | 0 | 0 |

Finally, the driving input of all reward was only at the DLPFC.

|  |  |
| --- | --- |
| Matrix C |  |
| VTA | 0 |
| NAcc | 0 |
| dlPFC | 1 |

Next we reconstruct the equations for each of the regions. Let’s call VTA: z1, NAcc: z2 and DLPFC: z3. Then we define *u*1 = high reward condition ($4) and *u*2 = high and low reward condition ($4 and $0). These are standard GLM regressors the length of the experiment, with sticks or miniblocks where events occur. They are not convolved.

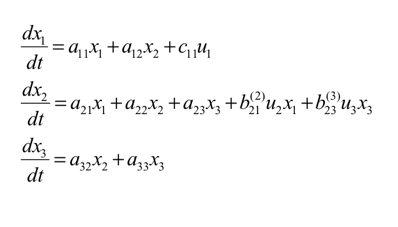
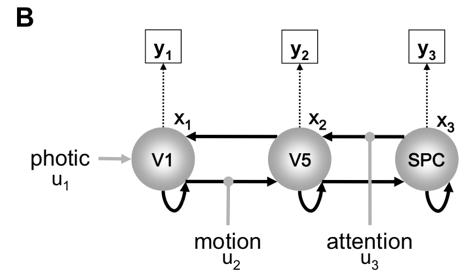
VTA: dz1/dt = A11*z*1 + A12*z*2 + A13*z*3 + B13*z*3*u*1

NAcc: dz2/dt = A21*z*1 + A22*z*2 + A23*z*3 + B23*z*3*u*1

dlPFC: dz3/dt = A31*z*1 + A32*z*2 + A33*z*3 + C3*u*2

These are the equations that would be specified to initiate a DCM. Once the DCM is estimated, you have a mean and standard deviation associated with each parameter. You can use these to determine whether each parameter is reliably above some threshold (e.g. 95% chance it is above 0) and report its mean or maximum a posteriori (MAP) estimate (these are equivalent under the assumptions used by DCM, more on this in the Bayesian section), and its variance or standard deviation. For example, the modulatory connection from DLPFC to VTA is significant, and its mean is B13=.20 and standard deviation is .07 (from table in DCM paper).

Here is another example taken from the attention to visual motion dataset publicly available on the SPM website:



Constructing a DCM

This section deals with how to transform a GLM design into a DCM. One of the first decisions you have to make as a modeler is how events will perturb the system. Typically, you are interested in how a particular experimental context (e.g. high reward motivation, high attentional load, perceptual interference) influences the connectivity between regions. In general, if your GLM results of main interest use the contrast of B>A, you are often interested in how B acts as a modulatory input. Though this is certainly not a rule and there may be scenarios in which this is not the case.

There are many ways to structure a DCM, but some designs are more interpretable than others. We will illustrate with an example. Assume you have 2 conditions (A and B) and 2 regions R1 and R2. From the GLM analysis, you observe an B related activation in R2 and you want to know whether it results from an increase in connection between R1 and R2 during B trials (a modulatory effect). The ‘correct’ way to do this is to include both A and B as a driving input to R1 and B as a modulatory input on the connection from R1 to R2. Then, by assessing this modulatory connection, you are testing whether a B related activation in R2 is better explained by 1) a nonspecific flow of information from R1 or 2) a selective increase in the connection between R1 and R2 during B. The ‘nonspecific flow” part comes from the driving input including all relevant conditions (A and B) entering at A and travelling to B via the intrinsic connection. By including only B as modulatory input, you are testing whether this B specific modulatory input adds explanatory power *over and above* the nonspecific flow of information. Thus, it would not make sense to set up models with 1) only A as a driving input and B as a modulatory input, or 2) include both A and B as both driving and modulatory inputs (think about how you could interpret the results from these scenarios). Also, I reiterate that for there to be a modulatory connection from A to B, there must be an intrinsic connection from A to B (but not the reverse).

In summary, all conditions of interest should enter somewhere as part of a driving input. Then, some of these conditions may separately act as modulatory inputs. (though there could be more than one driving input, For example, driving input 1 could be conditions A and B, and driving input 2 could be conditions C and D, and A,B,C, or D could act separately as modulatory inputs. This would be the typical setup, but for any design there will be a good deal of subjectivity and tuning to your particular question.)

The more orthogonal the driving and modulatory input, the better the modeling will work. See Klass Stephan:

*It is usually not a good idea to use the same input as a driving input to one region and as a modulator of an afferent connection originating in that region. The reason is that the conditional estimates of both parameters will be highly correlated and, given that the prior variance for driving inputs is higher than for modulatory ones, this will tend to over-estimate the driving parameters and under-estimate the modulatory parameters. In other words, your estimates are biased.*

Having driving and modulatory and driving inputs that are as different as possible will reduce this bias. However, since every modulatory input should also be included as part of a driving input, parameter estimates will be somewhat correlated. However, it is possible to assess the degree to which this is a problem by comparing the normalized posterior covariances (correlation) between parameters. Correlated parameters will be unreliable. However, even the hemodynamic parameters are correlated up to .3 (though more often in the range of -.1 to .1). One paper (http://www.ncbi.nlm.nih.gov/pubmed/20056151) that these correlations are reproducible across datasets for a given model and task, within and between different subject groups. Nonetheless, it is good practice to examine your correlations to ensure they are low (relative to those of the hemodynamic parameters). More on the mechanics on how to do this later.

Beyond constructing a single model, one must decide which of the enormous number of possible models to construct and how to compare them. The 10 simple rules paper does an excellent job of explaining how to do this, so I will not reiterate here.

The Bayesian Part

One of the more daunting aspects of DCM is the Bayesian framework. While the nitty-gritty is complicated and over my head, the general idea is actually not so hard to get and a few definitions greatly clarify the papers and SPM listserv posts.

Bayesian statistics is based on Bayes rule, a concept that allows you to invert conditional probabilities (the probability of x given that I know y -> the probability of y given that I know x):

In English this reads: the probability of y given x is equal to the probability of x given y, multiplied by the probability of y over the probability of x.

Further, we have an equation for p(x):

This says that the probability of x is equal to the sum of the probabilities of x given all the possible values of y, weighted by the probability of each of those values. We will illustrate with an example.

Let’s say you tested positive for a disease. You want to know the probability that you have the disease given that you tested positive. Let’s say that if you have the disease, there is a 95% chance you test positive (false negative rate = 5%), and if you don’t have the disease, there is a 5% chance you test positive (false positive rate =5%). Let’s also say that the disease has a 1% occurrence rate.

Note that the probability of testing positive p(+) (the denominator) is equal to the probability of testing positive given that you have the disease (.95) times the probability of having the disease (.01) plus the probability of testing positive given that you don’t have the disease (.05) multiplied by the probability of not having the disease (.99). This follows the equation above, but also makes intuitive sense because it accounts for all situations in which you test positive (you have the disease or you don’t), weighted by the probability of each of those situations.

With DCMs, we want to make inferences about model structure and parameters. This proceeds in a few steps:

1. We find a way to calculate the probability of parameters, given a specific model and some data
2. We find the parameters that maximize this probability
3. We use these “best” parameters to specify a particular model, and then use the probability of the model, given the data and those parameters, to compare models

Let’s call a given model m, the parameters of the model , and the data we are modeling (the BOLD timeseries) y. Then Bayes rule states:

So the probability of the parameters given the data and a model is equal to the probability of the data given the model at those parameters, times the probability of the parameters given the model, divided by the probability of the data given the model. Side note: for simplicity you can think about as referring to a single parameter (pretend we have a model with only one parameter). However, DCM estimates all parameters simultaneously (including the hemodynamic ones), so is really a vector of parameters. Time for some lingo used in Bayesian analysis and extensively in DCM papers:

**Posterior Probability**: ; the probability of the data given the data and the model. In other words, the strength of our belief in a parameter, taking into account the data

**Likelihood**: the probability that the observed data could be generated by a model with these parameters

**Prior Probability**: ); the probability of the parameter given the model. In other words, the strength of our belief in a parameters before we take into account the data

**Evidence**: the probability of the data given the model, across all possible values of the parameters

**Maximum A Posteriori Estimate** (**MAP**): The values of the parameters that maximize the posterior probability. Also called the “**conditional expectation**”

So, putting that together

The last piece is the equation of the evidence, which is a bit more complicated because parameters in DCMs can take on an infinite number of values. Thus, we replace the sum with an integral:

)\*p(

Basically, the evidence is the sum of the likelihoods given each possible value of the parameters, weighted by the probability that those parameters take on those values. We saw an example of this with the denominator in the disease example. Another example is a parameter β that takes as its value from first letter of the day of the week. Imagine that the mapping is {s,m,t,w,f} to {1,2,3,4,5}, so if the first letter is s, β =1, if the first letter is m, β =2, etc. Each of these values has a prior probability of {2/7,1/7/,2/7,1/7,1/7} based on what we know about weeks. The evidence for a model containing this parameter would be the sum of the likelihood the model given each of these values multiplied by their probabilities (\*2/7 +\*1/7 + \*2/7 +\*1/7 +\*1/7). In the case of continuous parameters, this sum becomes the integral of the likelihood times the prior over all possible parameter values.

Of note, for practical reasons, you usually take the log of this equation in order to solve it (specifically, logs let you represent the equation as sums and differences of quantities: log(posterior) = log(likelihood) + log(prior) – log(evidence). This removes the division and makes things computationally simpler). This is why you often hear terms like “log likelihood”: it is just the logarithm of the likelihood.

Everything in these equations is computed from the models and the data except for the priors. These represent our knowledge of what the parameters should “look like” before we take any data into account. For example, if region A and region B are completely disconnected in the model (based, for example, on anatomical constraints), then the prior probability of the parameter governing their connection being 0 is 1 (we are 100% sure this parameter is 0). DCM includes several additional constraints on the priors:

1. They are normally distributed (Gaussian). Therefore both their prior and posterior probabilities are fully described by the mean and standard deviation
2. All parameters have a prior mean of 0. This represents a very conservative approach saying basically: “unless the data can convince me otherwise, I don’t think any of these connections exist.” These are sometimes referred to as “shrinkage” priors.
3. Differential equations will sometimes “explode” to infinity, ie, predict exponentially increasing activity in each region (an example of such an equation (not a DCM): a\*x where a>0.) Models that do this are never good models of BOLD data. The variances of the priors are structured so that it is unlikely this happens.
4. Effects in the brain don’t last forever (eg: flashing a light doesn’t cause the visual cortex to stay active for the rest of the experiment). Thus, the variances are constrained such that the expected neuronal activity decay rate has an appropriate mean and variance (though decay rate isn’t itself a parameter, so its distribution isn’t constrained to be normal and it turns out to be skewed right). We prefer that neuronal activity will decay with a half-life of roughly 500 ms, falling in the range of 300 ms to 2 s.
5. The hemodynamic parameters have their own set of priors that have to do with assumptions about blood flow, et cetera, and I don’t know much about it.
6. You can set your own priors using things like DTI, but I don’t know how I feel about that. You basically assume that a thicker track makes it more likely a strong functional connection exists, and incorporate that assumption into the priors. Whether this is a just assumption is open-ended.

To make this clearer, we return to the example of a disease. In the case of the disease, we have a simple model of the disease with a prior on the likelihood of the disease in the population:

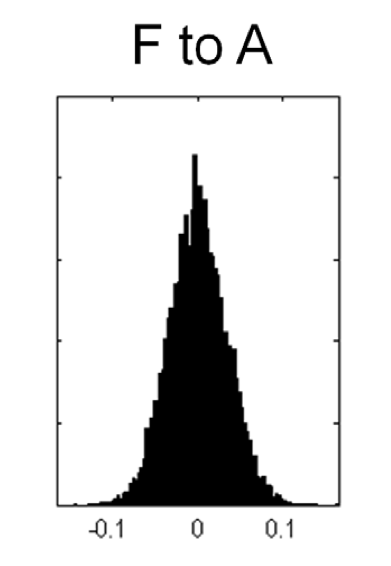
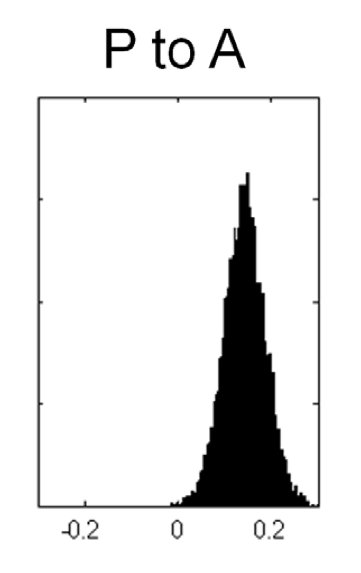
Here disease status is a discrete variable (either you have it or you don’t). Let’s say we had a variable that could take on any continuous value. For example, we could have someone draw a line randomly on a meter stick. Assume any value is equally likely to happen, thus making the prior flat (and completely uninformative). Basically, we think any value of this parameter in its range is equally likely.

In contrast, with DCM the priors are normally distributed, with their variance controlled by the above constraints. These distributions look like bell curves, and say that we think some values are much more likely than others. Importantly, we think that 0 is a much more likely value for any connection parameter than a number much greater or less than 0, which represents a conservative modeling approach.

Given all of this, these integrals are tremendously difficult to solve (recall that these equations are solved for all parameters at once), and as I understand it a big chunk of Bayesian statistics (and DCM papers) deals with how to approximate solutions to these equations for the evidence, and for finding the MAP estimates (the parameter values that maximize the posterior probability). You will see AIC (Akaike information criterion) or BIC (Bayesian information criterion) estimates reported in many papers using Bayesian modeling, including older DCM papers. The method used for DCM has changed, and the most current version uses “the negative free energy approximation to the log model evidence.” I don’t understand fully understand what this means, but it doesn’t really matter as long as you get that it’s an approximate solution to the problem of estimating the model evidence.

Once you have the means and standard deviations of the posterior probably distribution, you can test the probability that the parameter is greater than zero. Usually you pick some threshold, like 90% confidence that the parameter is greater than zero, but it is important to note that this is different than frequentist statistics (ie, traditional statistics, such as in a GLM) where you set a p-value for a hypothesis test. Rather, from the posterior, we can calculate the weight of our belief that a parameter is in a certain range and we must decide at what level we are suitably convinced.

For example, consider the posterior densities for the following two parameters:



The parameter governing the connection from P to A would likely be considered significant (a large portion, >95%, of its mass is above 0), whereas the parameter governing the connection from F to A would likely not be significant, since its mass is centered at 0 (~50% is greater than 0). Some papers assess significance at a group level by taking a t-test (or anova) of the MAP estimates across subjects. This is valid, but mixes different statistical frameworks, and as I understand it is more “graceful” to stay within the Bayesian framework.

Bayes Factors

Quick note on philosophy of modeling: there are no true models. If one existed, it would include billions of neurons, sodium channels, et cetera. We look for the model with the highest plausibility, or probability, given the data, and take into account both model fit and complexity.

The most straightforward metric for model comparison is the Bayes factor. Once you have the MAP estimates for the parameters of models M1 and M2, you can compute p(M1|D) and p(M2|D) using Bayes rule. If you shuffle around the equations, you can

show that:

This means that the ratio of the posteriors is proportional to the ratio of the evidences, and equal in the case that the prior probabilities of the models are the same (which they are for DCM). We define the Bayes factor as the ratio of the evidences

Bayes factors are intuitively an “odds ratio,” how much more likely one model is than another. In papers, one often sees the difference in log evidences, because:

A difference of log model evidences of 3 is often considered significant (BF=exp(3)), though this is an arbitrary threshold. This is the most basic model comparison method, and was indeed used by early versions of DCM and is commonly used in many contemporary publications. DCM uses a more sophisticated approach now, but the philosophy is the same.

As a side note, it is possible to model any data with a sufficiently complex model, but that doesn’t make it a “good” model in that it is probably fitting a lot of noise and will not generalize to different situations (eg: another subject’s brain). Ideally, a “good” model will fit the signal in the data and leave the noise unfit. (Importantly, DCM estimates the observation noise separately for each region, allowing for regional discrepancies in SNR). Luckily, the model evidence, by definition, takes into account both the accuracy and the complexity of the model. This is a great thing about Bayesian statistics. Note that prior probabilities must sum to 1 (the parameter has to take on some value). The more complex the model, the more the prior probabilities are spread out and the less informative the priors become (the less any particular parameter value has a substantial probability). So, if the data are equally accounted for by a complex and a simple model, we will likely have higher evidence for the simple model because the prior probabilities for those parameters will almost certainly be higher. Another way to say the same thing is that if model parameters are redundant, their posterior covariance will be high and this will penalize the model. Thus, there is a tradeoff between accuracy and complexity inherent in the definition of evidence for a model, and Bayes Factors will favor those models with the “best” tradeoff between model fit and complexity.

Model Comparison

A straightforward way to do model comparison at the group level is to compute a Bayes factor for each subject, and then simply multiply the Bayes factors of all the subjects to get a group Bayes factor. This corresponds to a fixed effect analysis (it assumes that the best model for every subject is identical). The current implementation of model comparison treats model identity as a random effect in the population. Thus, each subject’s data have some probability being best described by each model (or model family) considered. This part of DCM is well described by the 10 simple rules paper, and frankly I don’t understand the mathematics of the Dirichlet stuff, so I will give only a brief conceptual overview. This approach allows subjects to have probabilities associated with all the models, and they are different for different subjects. In addition to being random effects, this method is more robust to outliers than either the group Bayes factor or the frequentist approach of averaging model parameters. This approach gives you two metrics that give equivalent rankings of models:

**Expected posterior model probability**: how likely it is that a specific model generated the data of a randomly chosen subject

**Exceedance probability**: probability that one model is more likely than any other model, given the group data

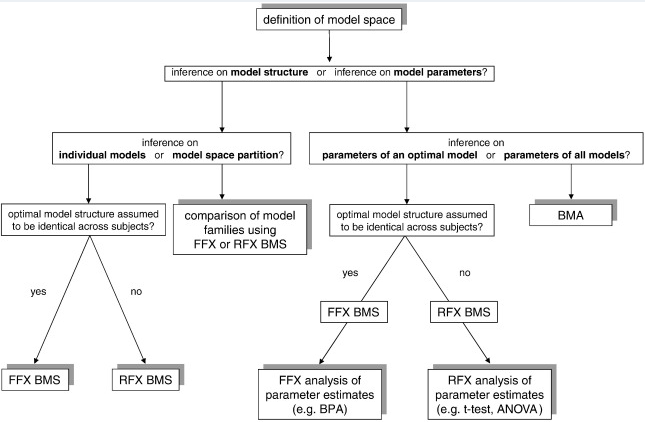
There are no accepted thresholds for significance for these quantities. If one is comparing 2 models or 2 families, it is often informative to take ratios of these quantities (eg: exceedence probability of M1 over the exceedance probability of M2).

This approach is very similar conceptually to Bayesian statistics at the level of a single model. In the same way as you specify priors in parameters, DCM specifies priors on the models. By default, all models (or families) are set to have equal prior probability. To include models (families) you think are improbable violates the assumptions. This is why you shouldn’t include models that, let’s say, have 2 connected regions and 1 disconnected one. This model is extremely unlikely given the connectivity of the brain. You could do it, but you would have to go into the code and lower the prior probability by hand. This is why you should constrain the model space with whatever knowledge you have. For example, if V1 is included in the model, driving input for visual cues should enter at V1 for all models. Visual input entering at V5 is implausible a priori, and violates the assumptions of model comparison.

Family model comparison is a particularly powerful approach. With large model spaces, the evidence becomes diluted over many models, and it becomes less likely that a “best” model will emerge. In general, finding a “best” model is also less interesting than testing specific hypotheses one has about models. Partitioning models into families that share a feature (eg: nonlinear versus linear models, or models with different driving input configurations) gets around the problem of dilution of model evidence and allows you to answer the specific questions you are interested in, pooling over features in which you are not interested. This approach gives you quantitative measures of features of interest of models, which is much more appealing that qualitative examination of the “best” single model. Thus, model space partitioning into families is both a more powerful and more interesting approach than searching for the “best” model.

Once you have a family that represents the data well, you can compute an “average” model using Bayesian model averaging. This procedure takes the average of each parameter across subjects and across models, weighted by each model’s evidence. For computational efficiency, models with low evidence are excluded from the average (models that are included are said to be in Occam’s window, i.e., sufficiently plausible and parsimonious). This approach pools information over a space of models and is, likely family level comparison, generally both more powerful and more interesting than searching for the “best” model.

I do not really discuss fixed effect analyses because they require very rigid assumptions that are unlikely to be justified for the types of systems our lab considers. This figure (from the 10 simple rules paper) should make complete sense before you go further. One qualification is that often one is interested in both model structure and model parameters (though I personally think model structure is generally more interesting). Nonetheless, it is possible to take both arms of this path for a single DCM project. For example, in the Ballard, Murty, et al paper, we used model space partitioning to perform random effects inference on model structure (where the driving input entered), then subsequently used Bayesian model averaging to examine the parameters in the winning family of models.



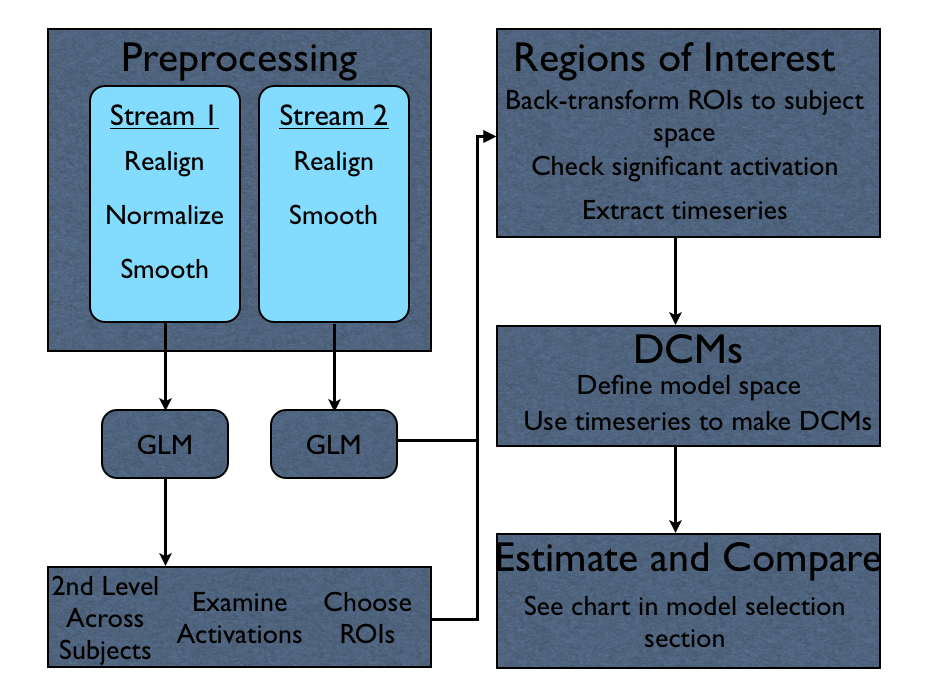
Advances

DCM has been extended to include nonlinear interactions (a region can influence the connection between two other regions in a context-sensitive way), stochastic models (these model the noise in the signal), two-state nodes (each region is comprised of an auto-excitatory and auto-inhibitory population), and slice-time correction (which models different slice times at the level of DCM, but not for interleaved). I have no direct experience with any of these methods, but they all seem pretty great.

Further, DCM10 implemented a number of changes, and it is important to report which DCM you use. In particular, the interpretation of the intrinsic parameter changes: “The mean-centering of inputs means that the interpretation of endogenous connection strengths (A matrix) changes. This is because the expansion point of the Taylor series is now the average of the input (u), therefore A no longer represents "baseline" connectivity (in the absence of perturbation), but the average connectivity (i.e., under average perturbation).” For a fuller list of changes, see: http://www.jiscmail.ac.uk/cgi-bin/wa.exe?A2=ind1011&L=SPM&P=R67576&I=-3&d=No+Match%3BMatch%3BMatches

Another issue with DCM10 is that models are apparently far more likely to “flatline”, or assume that the data are completely observation noise and return parameters that are at their prior means (0 for most of our parameters of interest.) DCM12, a beta release, is better at dealing with this problem, but still not as good as DCM8. This is still an open issue. For more details, see: http://www.jiscmail.ac.uk/cgi-bin/wa.exe?A2=ind1203&L=SPM&P=R98503&I=-3&d=No+Match%3BMatch%3BMatches

General overview of analysis stream



Above is a general schematic of how the processing stream works.

In stream 1, we do a normal GLM analysis (realign, normalize, smooth, first-level GLMs for each subject, second-level group models across subjects). DCMs offer alternative explanations of observed activations. Therefore, it is important to examine your group activations and decide which of those you are interested in modeling. Though it is technically possible to model a timeseries from a region not involved in the task, it does not make much sense. However, since DCM models multiple components of the task design to model the entire timeseries, it is perfectly acceptable to use different contrasts to define different ROIs (for example, regions showing anticipatory activation could be defined from a contrast of anticipation>outcome, and regions showing outcome activation could be defined by the reverse contrast. Both of these regions could be included in a single DCM that models both anticipation and outcome periods).

Once you have regions you are interested in modeling, you move to single-subject space. Since the analysis from here is in single-subject space, it is better to use un-normalized data (stream 2). Often, you may want to back-transform MNI space ROIs into individual subject space (such as a sphere centered around the group peak of an activation, or an atlas definition of an anatomical region). Then, you decide on an appropriate threshold for activation for individual subjects. On the listserv, Friston comments on this step:

*Remember, the regional response in each subject does not have to be  
significant at a corrected level and – strictly speaking – does not  
have to be significant at all. Having said this, you will get better  
results if DCM has something to explain and you choose your data from  
more significant voxels.*

How you chose to define this cluster is up to you. The more liberal you are, the more likely you include subjects for whome there is no effect and your DCMs will have mostly noise to model and this will weaken your power to detect effects across subjects. The more conservative you are, the more subjects you exclude, and you similarly lose power. In the Ballard, Murty et. al paper, we used a threshold of p<.05, uncorrected, 3 voxel extent in the midbrain and accumbens, and a 5 voxel extent in the cortex. Using alphasim to define a cluster extent threshold based on the size of the region is a promising approach we will implement in the future.

Once you have a list of subjects who have passed the threshold for all the ROIs in the DCM, you must decide how you want to extract the signal from the ROI. Like thresholding, this step will depend on the region and your personal preferences. In the Ballard and Murty paper, we extracted from the peak voxel. Many papers extract from all active voxels, or even all voxels within an anatomical ROI. The consideration of which approach to use is identical to that of standard ROI analyses.

Next, one extracts the principal eigenvariate of each region of interest. For a single voxel, this is identical to the mean. For multiple voxels, it is the timeseries that explains the largest variance shared by all voxels. In practice, it is often very similar to the mean. One often corrects the data for an F-contrast of effects of interest. This mean-centers the data, and corrects for nuisance variables (such as movement regressors, or multiple runs). One could also correct for task variables that you do not which to model with DCM. I prefer not to do this, as it seems to me to be a more conservative approach to leave the data as raw as possible. Klass Stephan has written on the listserv:

*If you deliberately zoom in on a particular set of experimental manipulations and disregard other manipulations, then the latter could be regarded as confounds and removed through adjusting your data during VOI extraction. I would be a little careful with this though since it questions the rationale of the experimental design that you chose in the first place. There may be very good reasons to do this but it always depends on the specific application. I'd say there is no general rule here.*

At this point, you are ready to define and estimate a DCM. It is often necessary to create (but not estimate) a new design matrix to provide the inputs to the DCM. Often, multiple conditions act together as a driving input, but separately as modulatory inputs (e.g. In the Ballard, Murty, et. Al paper, all monetary cues (5 and 0) were concatenated as a driving input, and the $5 cue was also modeled separately as a modulatory input). It is not necessary to estimate this GLM, it just needs to be specified so that the timing files are in a .mat structure that DCM knows how to interpret.

Finally you define your model space, create and estimate each model, and proceed with analysis according to the figure in the model comparison section.

Other Considerations

I made several decisions in this processing stream that had no clear right answer. I briefly go over these decisions and why I made them:

**Slice timing correction**

I chose not to use slice-timing correction because it adds an additional warp to the data. DCM can model slice-time effects explicitly, but not for interleaved sequences (which my data used). There is some debate about whether the degradation of the data by re-slicing is worth it. There is a section in the 2003 DCM paper showing it is quite robust to slice timing errors, as it has variables in the hemodynamic modeling that can “soak” up these differences. Further, I looked at a paper examining the effects of slice timing errors or GLM analyses (slice-timing effects and their correction for fMRI) and decdiced that for short TRs (1-2s), the advantage of slice-time correction seemed neglible.

**Multiple Runs**

The standard approach is to concatenate multiple runs into a single GLM and include nuisance regressors representing the runs (for n runs, include n-1 nuisance regressors). Some have argued that is better to create separate DCMs for separate runs and average the results (see listserv). This is less common in the literature, and I am not sure how it influences model comparison, so I prefer not to do it.

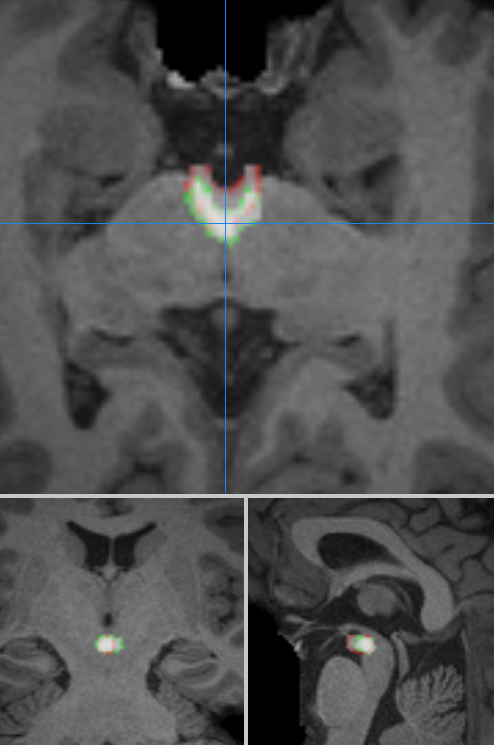
**Motion Effects**

You can model the motion effects as nuisance variables in the GLM. Further, when you extract the timeseries, you can correct for the nuisance effects of no-interest. I did this in the Ballard, Murty paper, but not in this analysis because the data were collected using Siemans automatic motion correction protocol. Thus, the additional motion correction is negligible, and I didn’t think it was necessary to include these regressors in the design matrix.

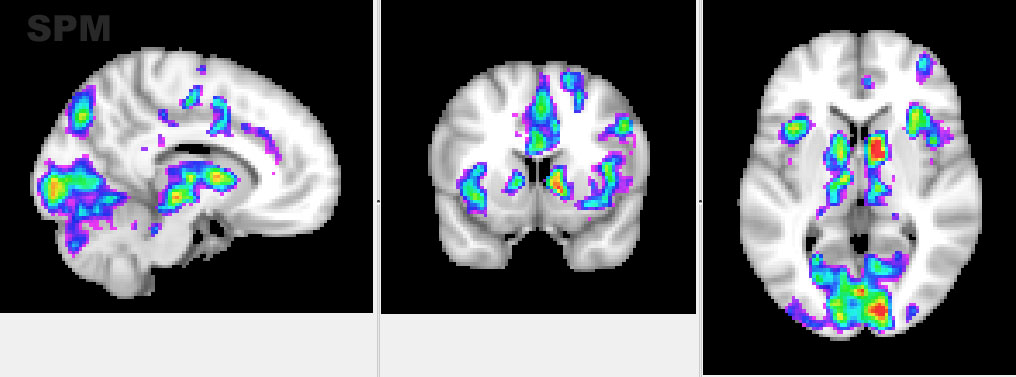
**Normalization in FSL**

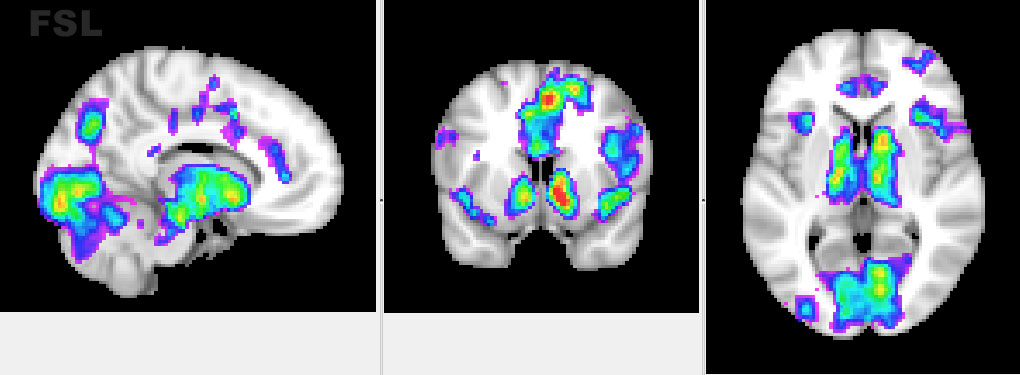
# Normalization in FSL using FNIRT, instead of in SPM, is a lot of work but I think it is worth it. One paper (Evaluation of 14 nonlinear deformation algorithms applied to human brain MRI registration) has shown that FNIRT is better than SPM’s normal normalization, but worse than SPM’s DARTEL toolbox, which combines segmentation with normalization. There are two main problems with DARTEL. First, as of now, there is no way to back-transform coordinates or ROIs. This severely limits its applicability for DCM, or many other analyses. Also, segmentation algorithms consider the midbrain to be white matter. Thus, in any analysis containing the midbrain as well as more typical grey matter regions, some will be preferentially normalized depending on whether you normalize white or grey matter.

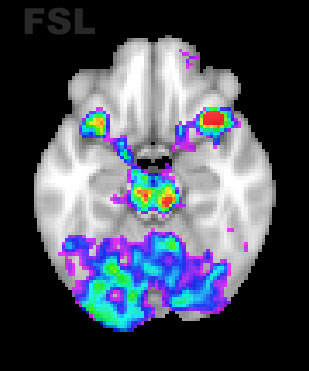
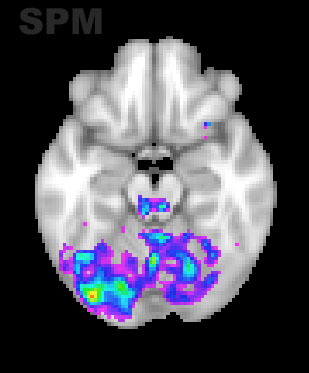
# Further, I examined our own data and found that FNIRT was considerably better in multiple respects. The data below labeled FSL were analyzed using the above DCM preprocessing stream for normalization (note that FSL normalization combines coregistration and normalization). The data labeled SPM were treated exactly the same except that they used the default SPM normalization routine. Below is an image of the back transform of the VTA probabilistic atlas into individual subject space. Green is from using FNIRT and red is from using SPM



I also compared group maps of the contrast of $5>$0 for the monetary incentive delay (MID) task. These images are on a common scale of 5-10, and the same slice:





Notice how, in particular for the accumbens, activations appear better localized and more significant. The same is true for the midbrain: 

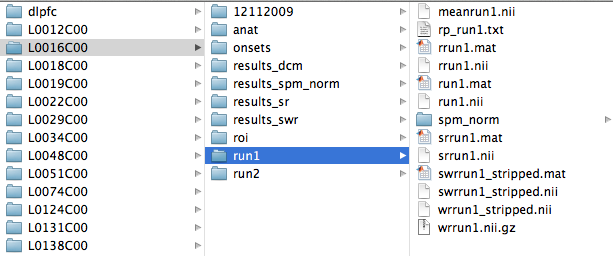
**Ordering of Preprocessing**

One difficulty with normalization in FSL is that FSL and SPM have different preprocessing step order. SPM normalizes unsmoothed data, the smoothes the normalized data, then calculates the statistical maps. FSL calculates normalization parameters on smoothed data, then calculates statistical maps, then applies these parameters to the resulting maps. I chose to preserve the SPM preprocessing stream as closely as possible. Thus, I use FSL to calculate and apply normalization to unsmoothed data, and then submit this to first-level statistical analysis in SPM.

One reason for doing this is that normalizing statistical maps means that the results will be extremely susceptible to errors in normalization, whereas, in SPM’s preprocessing, these effects could be minimized by smoothing and then calculating first-levels. Second, SPM labels voxels outside the brain as NaN in the statistical maps, and normalization gets confused by NaNs, so it would be necessary to pad the outside of the brain with 0 valued voxels. This is a lot of work, and since I have not found a strong explanation of why FSL does things this way (besides reducing file sizes), I see no reason to do it.

My intuition is that FSL calculates normalization parameters on smoothed data because the statistical maps that are ultimately normalized are generated from smoothed data. Since we do not do this in SPM, I chose to have FSL use un-smoothed data. It seems to me that this will be more effective for coregistering the functional and anatomical data, though I have not rigorously examined this.

**Directory Structure**



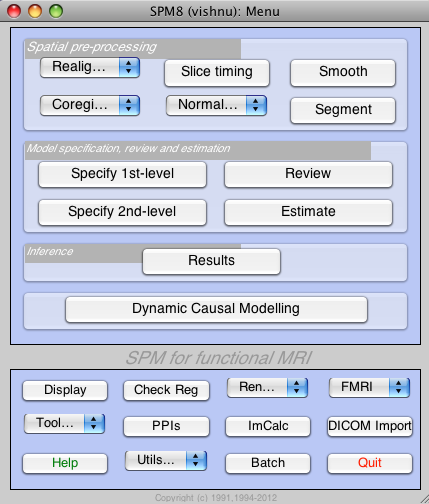
The leftmost column contains all of the subjects (and directories containing rois in MNI space, such as DLPFC). Within each subject’s directory, there is a directory containing her raw data (12112009), her anatomicals, the onset files, and her rois transformed into subject space. Additionally, each run gets its own folder and this is where preprocessed data go. There are three results directories (ignore results\_spm\_norm): results\_swr correspond to first-levels on data that has been realigned (r), normalized (w), and smoothed (s). These data will serve as input to group level analysis. results\_sr have first-levels on realigned (r) and smoothed (s) data. These data will let us find clusters of activations to extract from. Finally, results\_dcm will contain the first-level model with the dcm regressors, as well as the estimated DCMs.

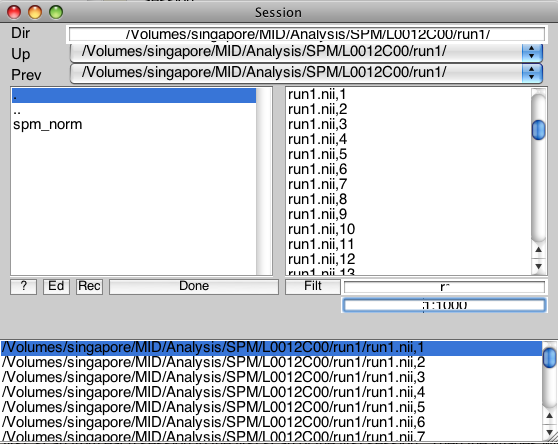
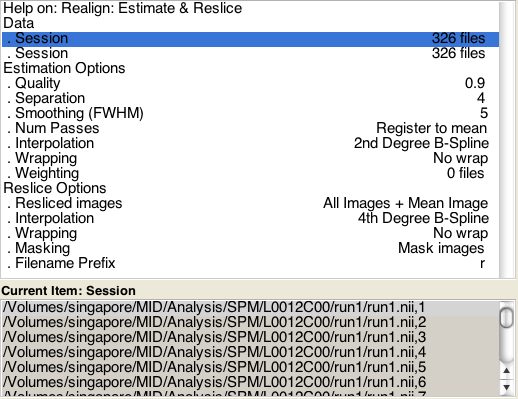
**Important few global points:**

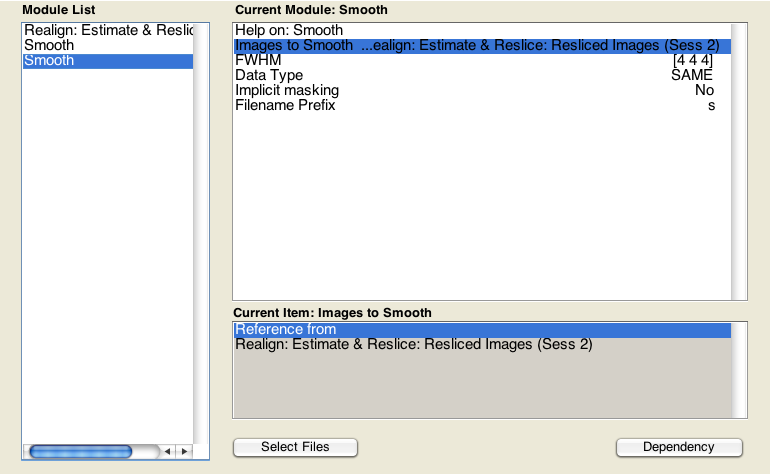
* This manual contains information for going through a practice analysis as well as information of general use. The latter will be italicized.
* This manual assumes that you are familiar with FSL, but not SPM
* All scripts with have a section in the beginning with User Settings. These are generally the only things you should have to adjust (path, file info, et cetera). However, for your own applications you may have to adapt the bodies of the scripts.
* SPM does not use .gz files, so all of your data will have to be unzipped. The data in the practice directory is unzipped.
* SPM keeps track of using prefixes. So you read from right to left to get the order of preprocessing. For example, swrrun1.nii indicates that the data have been realigned, then normalized then smoothed
  + R: realign (motion correction)
  + S: smooth
  + W: normalize
* The tutorial will use data from 3 subjects. The data and scripts are located at
  + /Volumes/adcock\_lab/main/resources/help\_and\_tutorials/dcm\_practice
* *This manual uses data and images as examples taken from the MID analysis from the Singapore project. These are healthy controls. If you are interested, you can find this full analysis at*
  + *Example data are located in the following directory:*
    - */Volumes/singapore/MID/Analysis/SPM*
  + *Scripts are all located in the following directory:*
    - */Volumes/singapore/MID/Scripts/analysis\_scripts/DCM\_workflow*
* We will be using spm, spmbatch, spmtools and wfu\_pickatlas. All are available as extensions to SPM and should be in the path or can be added at
  + /Volumes/adcock\_lab/main/resources/programming/matlab/spmtools/spmbatch.m
  + /Volumes/adcock\_lab/main/resources/programming/matlab/spmtools/
  + /Volumes/adcock\_lab/main/resources/programming/matlab/spm8/
  + /Volumes/adcock\_lab/main/resources/programming/matlab/WFU\_PickAtlas\_3.0.3\_Update\_for\_Matlab\_2011b/
  + *If not, you can download them again at:* [*http://www.fil.ion.ucl.ac.uk/spm/ext/*](http://www.fil.ion.ucl.ac.uk/spm/ext/)
  + *Note: spm8\_old is an older version of spm8 containing DCM8, which was used in the ballard, murty study*

Preprocessing: SPM and FSL

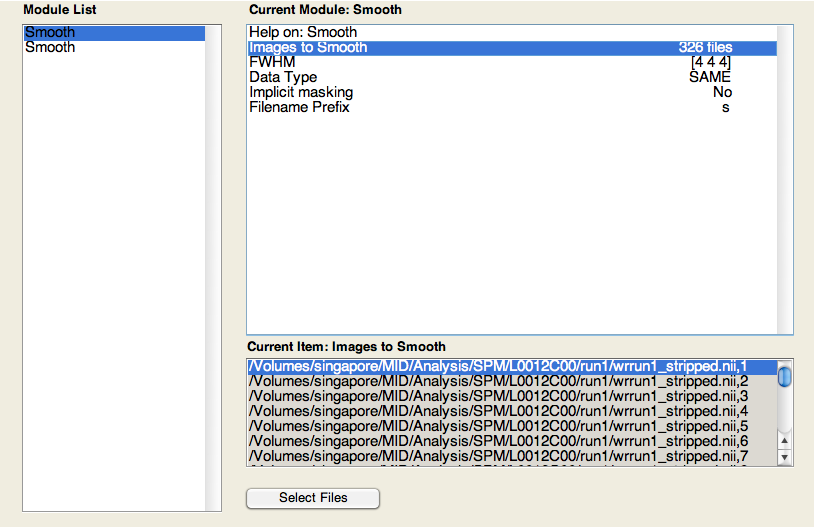
* ***QA tools should be used as in any fMRI study with the BIAC QA script or the QA info BIAC provides online.***
* ***Brain extraction and intensity normalization***
* *This has already been done for the tutorial*
* *SPM doesn’t do brain extraction, so this should be done in fsl*
* *I’m not sure if SPM does intensity normalization, but it doesn’t matter if it’s done twice and I think it’s necessary for brain extraction*
* *Make sure you keep the un-brain extracted image. Both images will be used for normalization. Name the brain extracted one full\_anat\_brain (FSL uses “head”, I used brain for my scripts. You can change it if you prefer). We are working outside the FSL GUI and using scripts with individual FSL commands, so it does not really matter.*
* *Copy the brain extracted functional data and both the brain extracted and the non-brain extracted anatomicals into the appropriate SPM directories.* 
  + *Make sure to unzip all images*
  + *An example of a script that does this is mover.py*
* **STREAM 2: Realign and Smooth (for the ROI timeseries extraction for DCM).**
  + Open Matlab, cd into the data directory, type spm fmri. You will see this window



* + Click on the batch button. This interface lets you do multiple processes in one script (such as realignment and smoothing).
  + *In the Singapore directory, you can open templates/preproc\_sr.mat for an example of how to set this up.*
  + Click SPM -> Spatial -> Realign -> Realign: Estimate and Reslice
  + Click Data-> New: Session. Add as many sessions as you have runs (in this case 2).
  + Click Session->Select Files. Navigate to where you keep the raw functional data (in this case run1.nii). We will just do this for 1 template subject. It is usually most convenient to just pick the first subject (ours is L0012C00)
  + You will need to split the data into however many 3D images you have. You can do this easily by
    - *Typing the first letter of the data you care about in the upper box (so it says r.\*), then click enter or Filt. This will filter the directory by the first letter*
    - Below, type 1:numvols. This will split the 4D data into 3D sections, I usually just type 1:1000 for convenience (1000 is just an arbitrary large number that’s easy to type), though of course if you have more than 1000 TRs you will need to type a bigger number.
    - Hold the shift key and select all your images. In this case there are 326. Repeat these steps for run2 in the next session.
    - *Note, if you make a mistake, you can select on the images in the bottom box and click them and they will disappear. However, they wont immediately reappear in the upper-right box. You will need to go back a directory by click the ‘.’ In the upper right box (highlighted in blue) and then re-enter the directory to refresh the listing of files.*
  + Leave all defaults. Your window should look like:
  + 
  + Next we will add smoothing. We will have to smooth each run separately. Click SPM -> spatial -> smooth. Do this twice.
  + Click images to smooth, then in the bottom right, click Dependency. Then click Realign: Estimate & Reslice: Resliced Images (Sess 1)
  + Change the FWHM to your desired smoothing kernel. (in our case [4 4 4]
  + Do the same thing for session 2.
  + Your window should now look like:



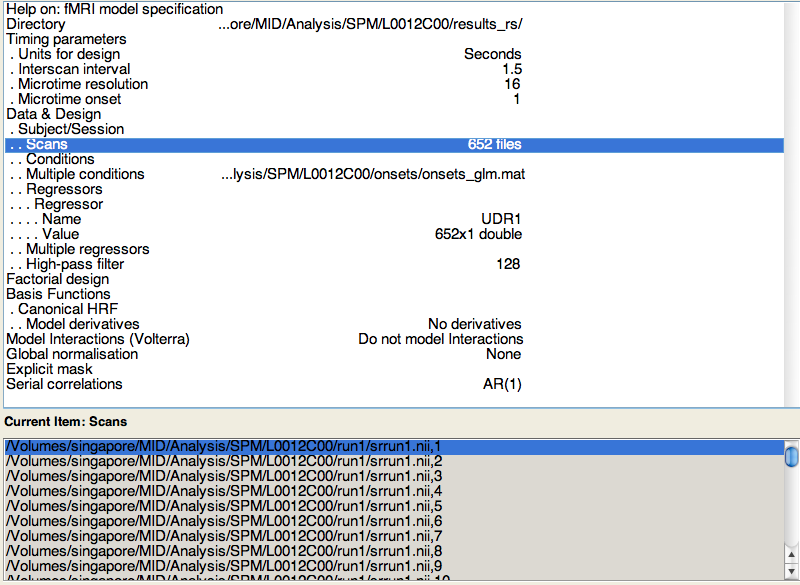
* + Save the batch in a templates folder (we call ours preproc\_sr)
  + Close the batch editor
  + We will now use spmbatch to make copies of this job for all the other subjects (making the appropriate path changes). They will be saved in a new folder called spmbatch\_jobs. Go to the matlab window and type: spmbatch(‘run’,’no’) (the ‘run’,’no’ part tells it not to run the jobs we are crearting. If you are not planning on using xgrid, you can leave this out and matlab will start serially running the jobs).
    - A window will appear asking you to select a template job. Select the .mat file you just created
    - A new window will appear asking for you to select a template subject directory. Select the template subject you used to set up the batch (the directory folder with the subject as the name).
    - A new window will appear asking for you to select subject directories. Select all subjects, including the template subject
    - In Finder, rename spmbatch\_jobs to something more informative, such as spmbatch\_jobs\_sr
  + You now have a job file for each subject. Next we will submit these jobs to xgrid.
* **Submitting Matlab Jobs to Xgrid**
  + This section will go over how to submit jobs (scripts in the format of .m files) to xgrid
  + The relevant scripts are: matlab\_xgrid,py, matlab\_xgrid\_submission.py, and template\_job\_file.m
  + *matlab\_xgrid\_submission.py is a superscript which will loop through either subjects in a subject list or all the jobs in a directory, and create a copy of matlab\_xgrid.py with the appropriate job variables. This will then be submitted to xgrid. This is very similar in principle to the superscripts we use for submitting normal FSL jobs.*
  + *Matlab\_xgrid.py will get variables specified by the superscript and renamed as a temporary, subject specific file. It is submitted to xgrid. Once at the host computer, it makes a job specific m file, opens matlab, executes the m file, closes matlab, then deletes the m file*
  + Template\_job\_file
    - *Runs within matlab*
    - *Adds paths, then uses spm\_jobman to execute the .mat file. However, it could do anything normally done in an m file. For our purposes, this will usually be running an SPM .mat job file*
  + To run all of your smooth and realign jobs.
    - Open matlab\_xgrid\_submission.py.
    - Make sure it loops through jobs by setting the loop\_variable to 1. (the spmbatch job names are kind of elaborate, so its easier to loops through these than a subject list)
    - Make sure that it is looping through the right spmbatch\_jobs directory
    - Then go to terminal, go the directory where the script is saved and run python matlab\_xgrid\_submission.py
  + By default, these scripts will output files useful for troubleshooting. However, the script comments explain how to automatically default these files if you’re lucky enough to have no troubleshooting worries.
    - template\_job\_file\_SUBNUM.m will have the exact job file that is submitted for each particular job or subject
    - testSUBNUM.txt will be a text file that contains the output you would normally see on the screen of a Matlab window. This is very useful to check the progress of the analysis and to easily access matlab errors.
* **STREAM I: Group Data: Realign, Normalize, Smooth**
  + Realignment (motion correction) is already done (from the previous step). These images are named, in our case, rrun1.nii and rrun2.nii (these are the default SPM prefixes)
  + For normalization, we will use 2 scripts. Fsl\_reg.py and submit\_reg.py
  + Submit\_fsl\_reg is a superscript that takes information about the directories and the loops through the subjects, each time making a copy of fsl\_reg.py with the appropriate variables inserted for submission to xgrid. This is very similar in principle to our FSL submission scripts.
  + Only open fsl\_reg if you need to change the paths of the anatomical files
  + *Fsl\_reg actually runs the normalization and coregistration with FNIRT and FLIRT, respectively. The majority of it was copied from a log of a typical fsl FNIRT job.* 
    - *The script makes a /reg directory like fsl does, and outputs most of what fsl normally does (I cut out the .png pictures)*
    - *Make sure the correct files are getting copied in (meanrun1 should be the copied in as example func because everything is realigned to this image). Make sure full\_anat, full\_anat func, and the template images are what you want them to be (lines 10-23)*
    - *The following lines (up to 64) are copied in from an fsl log file (though I have added comments and checked to make sure everything is doing what it should)*
    - *Lines 64-68 apply the transform to rrun1 and rrun2, naming them wrrun1 and wrrun2*
    - *One problem with normalizing the functional images is that it makes huge files. When SPM does this, it crops the files substantially (so much so that often there is loss of brain at the edges). It seems that if the files are bigger than 1gb, SPM is unable to open them and crashes. To get around this, we crop some of the non-brain voxels to reduce the image size.* 
      * *We use fslroi. I just take 5 slices off the end of the image in all 3 dimensions. Since this is MNI space, I think this should present no problems for any other datesets. However, I would check a few subjects to make sure this isn’t cropping brain.*
      * *For some reason, Mango gets confused by the cropping and shows the images displaced, but mricron, fslview, and SPM all understand the cropped images. These images are name wrrun1\_stripped*
    - *Finally, we unzip the images to appease SPM*
  + Make sure the correct subject list is specified and run submit\_fsl\_reg.py in terminal
  + Smooth the images
    - Use SPM in the same was as from the realign and smooth section (leaving out realignment) to set up a smooth batch.
      * Click SPM -> spatial -> smooth. Do this twice.
      * Click images to smooth, then in the bottom right. Select the wrrun1\_stripped.nii images
        + Remember to splice into 3d images (type 1:1000 in the bottom input box and press enter)
        + *To only see images with a specific prefix, typing the first letter of the data you care about in the upper box (so it says w.\*), then click enter or Filt. This will filter the directory by the first letter*
      * Change the FWHM to your desired smoothing kernel. (in our case [4 4 4]
      * Do the same thing for session 2.



* + - Run using spmbatch and matlab\_xgrid\_submission.py as before (Make sure that it is looping through the right spmbatch\_jobs directory)

GLM within-subject analysis in SPM

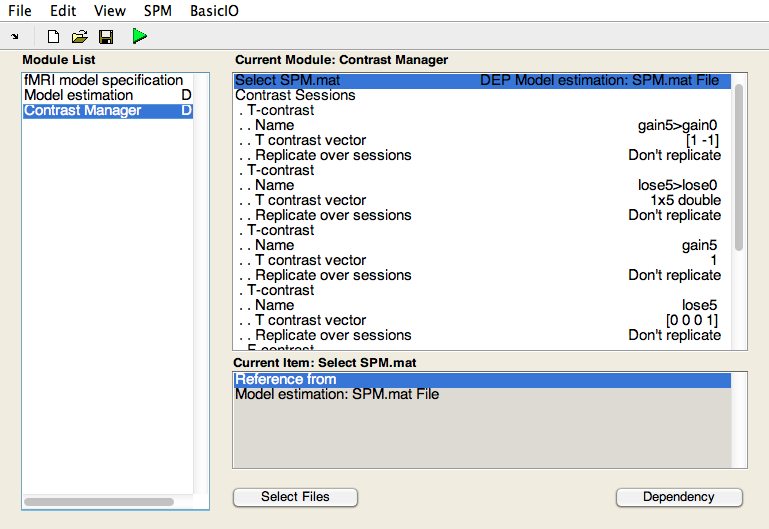
* We need to run 2 sets of first levels that are identical except for the input data
  + One on the realigned and smoothed data (srrun1.nii) for use in ROI definition and thresholding
  + One on the realigned, normalized, and smoothed data (swrrun1\_stripped.nii) for between-subject analysis (standard fMRI analysis)
* First, we need to specify the condition names, onsets, and durations in a .mat file that SPM can understand (SPM does not take 3 column text files like fsl)
  + *An example onset .mat file named onsets\_glm.mat can be found in any of the subject’s onset directories in the singapore directory where the full dataset is*
    - *This file is a data structure containing 3 cells: names, onsets, and durations*
    - *names{1} corresponds to onsets{1} and durations{1}, as for all other indices*
  + An example of a script that makes these files from the 3 column text files is called make\_glm\_onset\_files.m
    - If onset files in the 3 column format from FSL are in each subject’s onsets directory, this will make a .mat file
    - Make sure to include the number of scans and TR, as this is important for concatenating the onset files. IE, we are modeling run2 as starting at the moment in time that run1 ends
    - It’s important that the data path for this script have a ‘/’ at the end: eg path= ‘/Volumes/adcock\_lab/’
* We will illustrate setting up first levels with the realigned and smoothed data
  + Design Specification
    - Open the batch editor
    - Click SPM->Stats->fMRI model specification
    - Double click “Directory” and click a folder for the results to go.
      * You will have to make this folder in advance (just type mkdir in the matlab command line).
      * I call the folder results\_rs.
      * Note: In the window that pops up to select a directory, to select a directory you are already in, you can click the ./ on the right side. Alternatively, you can go back a directory (click ..) on the left side, and then click the directory on the right.
    - Specify the units for design (we use seconds the way we specified the onset files) and interscan interval (this is the TR, don’t know why they use this lingo. Enter the value for the TR)
    - Double-click “Data & Design”
    - Double click “Scans”
    - Select the data
      * Select the srrun1.nii files
      * Remember to splice into 3d images (type 1:1000 in the bottom input box and press enter)
      * Don’t click done!
      * Select the srrun2.nii files (since we are concatenating runs)
    - The Scans level should now contain the sr files from run1 and run2.
    - Double –click “Multiple conditions” and select the onsets\_glm.mat file
    - We need to make a nuisance regressor representing the concatenation of runs
      * Double-click Regressors
      * Name it something like “UDR1” (user designed regressor 1)
      * In value, input an array of ones for run1 and 0s for run2. The whole array should be the length of run1 and run2 combined
        + You can get this by typing [zeros(326,1) zeros(326,1)] in the matlab command window, and simply copy and pasting the resulting vector into the SPM batch input box
      * *If you had more than 2 runs, you would add more regressors. If n is the number of runs, we ran n-1 nuisance regressors*
    - *If you wanted to include realignment parameters (I don’t in this example), you would click multiple regressors and select a file of the realignment parameters (concatenated rp\_run1.txt and rp\_run2.txt)*
    - Leave all other options as defaults
      * *Importantly, we do not bother to model temporal derivatives as the DCM will implicitly model these*
    - Your window should look like



* + Model Estimation
    - Click SPM->Stats->Model Estimation
    - Click Dependency, then select: “fMRI Model Specification: SPM.mat file”
    - Leave estimation as classical
  + Contrasts (this step is analogous to what you do in FSL)
    - Click SPM->Stats->Contrast Manager
    - Click “Select SPM.mat”, then click Dependency and select:” Model Estimation: SPM.mat file”
    - Click “Contrast Sessions”, then click “new T-contrast”
    - Name the contrast and input a vector just as you would for FSL (eg, cond3>cond1 would be -1 0 1)
      * SPM will “pad” the right with zeros. For example, you don’t have to write [1 -1 0 0 0 0 ], [1 -1] will work fine.
    - Include all contrasts of interest.
      * The names will be used if you load the results through SPM’s results tool.
      * *However, it is useful to know that SPM names its files in order. For example. the first t-contrast will be saved as spmT\_001.img.*
    - Make sure you include an F contrast for effects of interest. This will be used to mean correct the data for extraction, plus for correction for nuisance effects like motion and (if you chose), conditions of no interest. In essence, this removes the effects you don’t care about from the timeseries for DCM. I think this is roughly (or perhaps, exactly) equivalent to taking the residuals of a model that includes only nuisance effects, and then doing DCM on the residuals. For n columns, this will be an n by n matrix, where each column “of interest” gets a single 1 in a different row than all the other effects of interest, and effects of no interest get all 0s. For 3 effects of interest, and 1 nuisance regressor (e.g. for concatenating runs), the matrix would look like:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 |

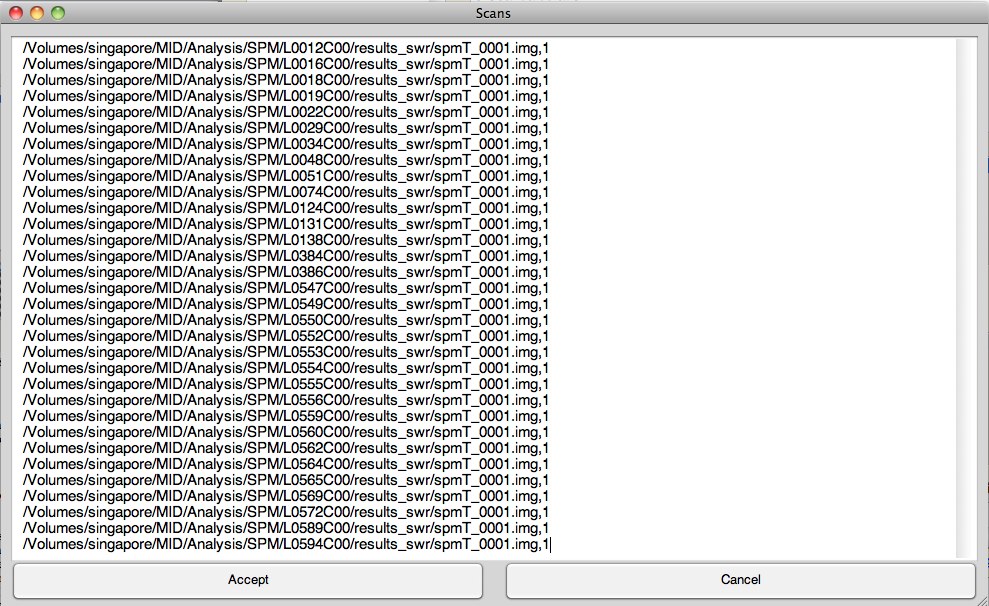
* + The batch should now look like:



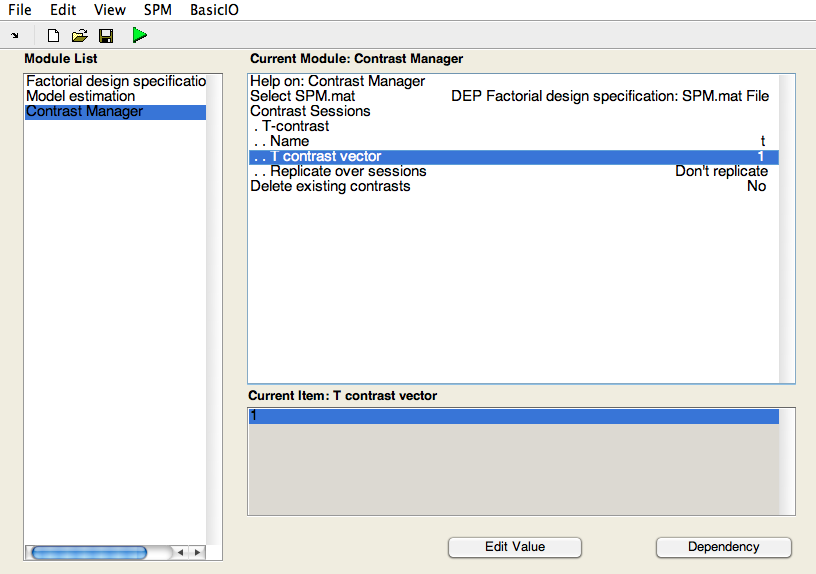
* + Save the batch something like first\_level\_sr
  + Run using spmbatch and matlab\_xgrid\_submission (Make sure that it is looping through the right spmbatch\_jobs directory)
* Do the same with the normalized data, being sure to make a different results directory and to load in the swrrun1\_stripped data. Everything else should be identical.
* *You can view the results using the results button and loading the SPM.mat file.*

GLM group level analysis in SPM

* Now we perform what in SPM is called a second level analysis (note: we do not do a second level analysis as FSL does, because we concatenated runs into a single GLM). This analysis is across subjects.
* Design specification
  + Open the batch editor
  + Click SPM->Stats->Factorial Design Specification
  + Double click “Directory” and name a folder for the results to go. I call the folder results\_norm
  + Under Design->One-sample t-test, double click “Scans”.
  + In the window that opens, click “Ed”. This opens a text editor window for you to type in scans.
    - You give it the spmT image from each subject for your contrast of interest.
    - If you are interested in multiple contrasts, you will have to do this separately for each contrast. In this case, I am interested in $5>$0, which is spmT\_001 (the first contrast I specified).
    - I have a little script that will generate this list for you (output to the terminal window), since its irritating to copy and paste and edit for each subject. It is called make\_lists.py



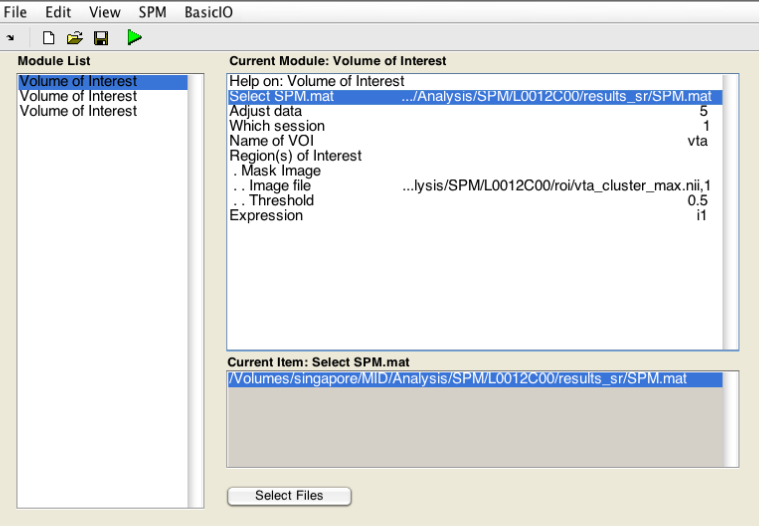
* + - Click “Accept”
    - Leave the other presets
* Model Estimation
  + Click SPM->Stats->Model Estimation
  + Click Dependency, then select: “Factorial Design Specification: SPM.mat file”
  + Leave estimation as classical
* Contrasts (this step is analogous to what you do in FSL)
  + Click SPM->Stats->Contrast Manager
  + Click “Select SPM.mat”, then click Dependency and select:” Model Estimation: SPM.mat file”
  + Click “Contrast Sessions”, then click “new T-contrast”
  + Double click “Name” and insert a name
  + Double click “T contrast vector”, input 1, and select “OK”
* The batch should look like:



* Save the batch something like “second\_level”
* Click the green arrow to run
* You can look at the results by clicking the Results button and loading the SPM.mat file that is generated or by opening the SPMT file in a viewer like MriCron

Defining ROIs

* Once you have looked at your group maps and decided which regions to include in your DCM, you need to extract the principal eigenvariate of the region of interest
* Sometimes, you may have anatomical boundaries in individual subject space. More often, you have landmarks or anatomical masks in group space and you need to back-transform them using the normalization parameters
  + You will use 2 scripts: transform\_rois.py and submit\_transform\_rois.py
    - Submit\_transform\_rois.py takes as input the paths to ROIs and the names you want to give them. Then it loops through subjects and makes a copy of transform\_rois.py to submit to xgrid
    - *Transform\_rois.py applies the inverse nonlinear warps to the ROIs to transform them to functional and to anatomical space. We will be working in functional space, but it is useful to also check the warp to anatomical space to make sure everything looks good*
    - Submit the jobs to xgrid, and this will output warped ROIs into each subject’s anat/reg/ directory
* The next step is to see which subjects meet some criteria for inclusion in the DCM analysis. Typically this will include some cluster extent threshold at a certain p-value
  + *Since we are working with T stat images, you will need to convert your p-value threshold into a t-stat threshold*
    - *T stats are a function of p value and degrees of freedom. You can convert between them using online applications such as at:* [*http://www.stat.tamu.edu/~west/applets/tdemo.html*](http://www.stat.tamu.edu/~west/applets/tdemo.html)
    - *Degrees of freedom*
      * *For multiple regression, degrees of freedom are normally equal to number of data points – number of columns in the design matrix – 1*
      * *For fMRI data, degrees of freedom end up being a little less due to either or both (I’m not sure) that:*
        + *Scans close in time are non-independent*
        + *Assumptions about modeling fMRI data are included as ‘hidden’ columns*
        + *See for details:* [*http://users.fmrib.ox.ac.uk/~stuart/thesis/chapter\_6/section6\_4.html*](http://users.fmrib.ox.ac.uk/~stuart/thesis/chapter_6/section6_4.html)
      * *In any case, the “true” degrees of freedom can be found by loading the SPM.mat for that design matrix and typing SPM.xX.erdf.*
  + We will use a script called get\_cluster.py. Run this script for each ROI. You need to input:
    - The name of the results directory
    - The name of the stats image you are looking at
    - The t threshold for stats images
    - The cluster extent threshold
    - The name of the roi
    - The probability threshold if you are using probabilistic atlases
  + *The script takes the mask in subject space (done above), thresholds it, binarizes it, and multiplies it by the stats image. The resulting image is submitted to fsl’s cluster command with the t threshold. This command outputs a list of non-contiguous clusters significant at this threshold, as well as some info about them. The script will output a list of offending subjects to be excluded from further analysis.*
    - You should make a new subject list called ‘roi\_fails.txt’ or something similar with the offending subjects. The scripts after this point will use this text file to exclude those subjects from future analyses.
* Once you have decided which subjects to include, you need to decide from which mask you should extract the timeseries.
  + *There are two opposing concerns here:*
    - *The more voxels you include, the less noisy the resulting signal*
    - *The more voxels you include, the less likely you are extracting a ‘reasonably’ functionally homogenous signal*
  + For my purposes, I decided to include all voxels within the peak cluster identified by the cluster thresholding above (p<.05, 3 voxel extent) contained within the ROI boundaries. The peak cluster is the cluster with the voxel containing the highest t-stat, though I noticed in my data that this peak cluster was also always the cluster with the greatest extent.
  + A script, make\_masks.py, creates a mask of the peak cluster using the masked SPMT image from get\_clusters and various fsl tools.
* Next, you need to extract the principal eigenvariate from the ROIs for the DCM
  + *Of note, this ROI may not be the same you use to threshold. It is perfectly reasonable to threshold by looking for supra-threshold clusters in an entire region R1, and then only extract from the peak voxel, or the all supra-threshold voxels.*
  + Given this, you will need to create an ROI mask.
    - Open SPM’s Batch interface
    - Click SPM->Util->Volume of Interest
    - Select the SPM.mat from the results\_sr folder (for just the 1st person)
    - For “adjust data,” type the index of the F-contrast for effects of interest. Important: even if it is the first F contrast, SPM keeps track of them by the index in which they were created. Thus, if you first specified 2 t-contrasts, and then an F-contrast, its index would be 3 (you can check the index by looking in the results\_sr folder and looking at the extension of the T or F.hdr or .img files).
    - Enter “1” for which session (we concatenated sessions, so there is only one)
    - Type in the name of the VOI (nacc or whatever you want to call it)
    - Select the mask image (roi\_cluster\_max.nii) if you used the make\_masks.py script (choose this for the regions of interest- click new mask image – then choose your roi in the mask file)
    - Threshold does not matter, any value is fine
    - For expression just put “i1” (this means use image 1). *If you had put multiple images in the mask section you could do arithmetic operations on them here. I would personally prefer to do this with fslmaths.*
    - Repeat for each volume of interest (spm – util – volume of interest)
    - Save something like ‘extract\_vois” (save in the templates folder and close batch editor)
      * use spmbatch (from command line of matlab – type spmbatch – then follow the prompts and select appropriate files) and matlab\_xgrid\_submission to run for all subjects (Make sure that it is looping through the right spmbatch\_jobs directory)
      * Note: you may have to modify your submission scripts at this point to skip the roi\_fail subjects. However, worst-case scenario is you submit these extra jobs to xgrid and they fail because there are no masks.



Generate and Estimate DCMs

* Next, specify a GLM containing the appropriate onset files for your DCM. *This may include concatenating multiple conditions into extra regressors for driving inputs. It will also simplify things if you simply remove conditions that won’t act as inputs to the DCM.*
  + make\_dcm\_onset\_files.m can be modified to do this. As it stands, this creates a regressor for $5 and $0 events combined to act as a driving input and a $5 regressor as a modulatory input. This makes a .mat file in the onsets directory called onsets\_dcm.mat
    - It’s important that the data path for this script have a ‘/’ at the end: eg path= ‘/Volumes/adcock\_lab/’
  + Use the batch interface to specify models as with the other first levels (include the TR and units for design. No need to include regressors of no interest like concatenation regressors). There is also no need to estimate this model or define contrasts. This just stores information about the design in a way SPM understands to be used when constructing the DCMs
    - This is fast, so you can just use spmbatch without the (‘run’,’no’) flag and run the batch jobs serially
* Open make\_base\_dcm.m. This script was modified from one available on the listserv. I have commented what I understand, and the rest I have checked simply by comparing the output of the script and the output of the GUI
  + This script will make a base\_dcm.mat model for each subject containing information about the inputs and VOIs, but with empty A, B, and C matrices
  + *Specify the ROIs, the SPM.mat, and the TE, as well as examine additional comments.*
  + *You will have to tinker with it depending on the number of inputs you have and what index they are in the design matrix. I strongly recommend making a model via the GUI as well and checking whether they correspond.*
* Make\_model\_space.m is a script that will copy the base\_dcm, inputing the appropriate A,B, and C values over the whole model space
  + This script will need to be edited extensively to suit your model space
  + Often you will want to do all possible permutations along a given dimension (such as all combinations of driving input). The script is set up to do this. The equations seem more complicated than they are. I commented them, but I recommend taking chunks of it and running it in matlab and I think it will make sense quickly. The script is fast for 1 subject, so play around with it and load the DCMs and look in the A, B, and C files to see if they are what you expect
  + *Note: You read the matrices “from column to row”. Thus, A(1,3) will give you the connection from region 3 to region 1.*
* Use xgrid\_matlab\_dcm\_submission and the spm\_dcm\_estimate command to estimate all the dcms
  + This will set the path to SPM12 for DCM estimation. You may have to set it back later for other applications.
* It is always a good idea to check out your DCMs to make sure they are fitting OK
  + It is useful to type “help spm\_dcm\_estimate” to help interpret the results outputted by the estimation
  + Make sure this directory is in your path:
    - /Volumes/adcock\_lab/main/resources/programming/matlab/spm\_dcm\_explore
  + spm\_dcm\_fmri\_check(‘model.mat’) gives a quick check showing the percent of variance explained and the number of estimable parameters
    - Friston wrote this script and said that DCMs should predict 10% of the variance. However, I’ve noticed in my own data and in other posts on the lists that DCMs often perform worse than 10%. However, they should do better than 0 or 1%, as this usually indicates that all the parameters have shrunk towards their prior expectation of 0 (the DCM flatlined, or concluded that the data were completely observation noise)
  + spm\_dcm\_explore(‘model.mat’) gives a GUI interface for viewing the results of the modeling.

Model Comparison

* Now that all models have been estimated, all that is left is to compare the models.
* Open the SPM batch editor and select SPM->Stats->Bayesian Model Selection->BMS:DM
* Select an appropriate results directory
* Either select each subject’s DCMs in the data directory or create a .mat file for the model space
  + If you are running a large number of models, it is inefficient to do this by hand. I have written 2 scripts, make\_model\_space\_mat.mat and make\_faily\_mat.mat. These make the .mat files from the data that SPM wants. For details on what these .mat files should look like, just click on their corresponding options in the Batch interface to read about the mat files.
* Select inference method (probably RFX)
* Input families if you are doing family level inference. Each family takes a vector with length equal to the number of models. Models should be indexed with a 1 if the belong to each family and a 0 otherwise.
* Select whether you want to perform Bayesian model averaging (this will automatically do BMA on the winning family)
* Let it verify data identity (just as a safe check)
* Click Run!
* You may want to look at the BMSmat.pdf file in this same directory for a detailed description of each of the fields in the outputted file
* You can also do Bayesian model averaging afterwards by selecting Dynamic Causal Modeling on the SPM GUI, then click compare, and load the BMS.mat file and select a family to average within

Reporting Results

* Always report which version of DCM used, how you chose your model space, whether you used RFX or FFX, how you set up your model families, and both the posterior means and posterior variances (or standard deviations) of the parameters of your reported model (whether this is a winning model or a model from BMA).
* You will have to decide a significance level for reporting connections (90% is often used, though this is flexible).
* Report both the expected posterior probability and the exceedance probability, as well as the ratio for each of these between the best and second best model or family
* To examine posterior correlations between parameters, or the extent to which effects explained by one parameter can be equally well explained by another (1 would indicate that they are completely correlated and its impossible to accurately estimate them separately, whereas 0 would indicate that they are entirely independent and changing one parameter would have no influence on the other)
  + You can see the full matrix by using the spm\_dcm\_explore function
  + Getting specific values requires normalizing the posterior covariance matrix DCM.Cp
    - First, compute the (posterior) standard deviation for each parameter (this is the square root of its diagonal entry in DCM.Cp
    - Second, divide each element in DCM.Cp by the product of the (posterior) standard deviations of the respective parameters.
    - Getting the Order of parameters is a bit tricky, but they are in the same order as in the DCM.Ep vector, which holds their posterior means, so you can use that to intuit which parameter is at which index.

Troubleshooting

* Matlab Xgrid
* There was an error with running SPM on the 10.6.8 machines. They were unable to do any of the unix commands, such as ls. It would output the following error:
  + { Warning: MATLAB was unable to open a pseudo-tty: Operation not permitted [1,1]  
    The unix() and ! commands will not work in this MATLAB session.  Other  
    commands which depend upon unix() and ! will also fail.  Your system may  
    be running low on resources.  If the problem persists after a reboot,  
    check with your system administrator and confirm that your pty subsystem  
    is properly configured}
* SPM is unable to use the system(‘hostname’) command to get the name of the computer. Our python script can tell the computer name though. So in we use the matlab\_xgrid script to pass the computer name to the template\_job\_file, which sets it as a global variable host\_name.
* Then, we go into the script that is giving the error (spm\_platform) and comment out lines 211-223. These lines use matlab tools to get the host name. We simply add 2 lines
  + global host\_name
  + PLATFORM.host = host\_name
* This is sort of a “band-aid” solution. I still don’t know what’s going on. But spm\_platform gets the computer name and it seems to be happy with that.