Dynamic Causal Modelling of fMRI data in SPM12

Pranay Yadav[1*] & Richard N Henson[1,2]

[1]MRC Cognition & Brain Sciences Unit, University of Cambridge, Cambridge, UK

[2]Department of Psychiatry, University of Cambridge, Cambridge, UK

*** Correspondence:**
Corresponding Author
pranay.yadav@mrc-cbu.cam.ac.uk

Keywords: MEG, EEG, fMRI, multimodal, fusion, SPM, inversion, faces

Abstract

Dynamic Causal Modelling (DCM) is a widely used method for inferring effective connectivity from various kinds of neuroimaging data. This tutorial demonstrates a step-by-step walkthrough for using DCM to investigate group-level effective connectivity from a publicly available open-access fMRI dataset for face processing from 16 subjects. We illustrate a reproducible analysis pipeline that makes use of a hierarchical Bayesian framework called Parametric Empirical Bayes (PEB) to characterize inter-individual variability in neural circuitry. At the group level, we show various approaches for performing testing focused hypotheses on the estimated connectivity using Bayesian model comparison.

1. Introduction

This paper is a tutorial for Dynamic Causal Modelling (DCM) of fMRI data. DCM is a method for inferring effective connectivity between brain regions from neuroimaging data, and is part of the Statistical Parametric Mapping (SPM) free academic software (https://www.fil.ion.ucl.ac.uk/spm/), which is implemented in Matlab (The MathWorks Inc., 2018). We focus on basic DCM specification, estimation, Bayesian model reduction and Bayesian model comparison, using the recent Parametric Empirical Bayesian (PEB) framework for group-level inference across multiple subjects. Together with its companion document describing DCM of MEG data, this document extends the DCM PEB tutorial by Zeidman and colleagues (Zeidman, Jafarian, Corbin, *et al.*, 2019; Zeidman, Jafarian, Seghier, *et al.*, 2019) to a multimodal dataset, and illustrates other features of DCM and PEB (https://github.com/pzeidman/dcm-peb-example).

The dataset contains fMRI and MEG+EEG data on 16 subjects from a face-processing paradigm described in Wakeman & Henson (2015). The raw data in BIDS format are available on OpenNeuro (https://openneuro.org/datasets/ds000117). This tutorial continues a previous tutorial on the same dataset (Henson *et al.*, 2019), which illustrated basic pre-processing and source localisation of MEG/EEG/fMRI data in SPM. Here we assume this pre-processing has already been done, though you can download the pre-processed data as described below.

We describe practical steps using 1) SPM's graphical user interface (GUI), 2) its "batch" interface for linear pipeline creation and finally 3) "scripting" in MATLAB for (parallelised) loops across subjects. We use version 12.5 of DCM in version 12 of SPM. The paper is organised into sections with a brief theoretical background followed by a detailed step-by-step walkthrough. The background is only brief because we refer to previous published papers, many of which are available from the SPM website: https://www.fil.ion.ucl.ac.uk/spm/doc/biblio/. We do not provide a full tour of all the available options in SPM for M/EEG, which is already present in Litvak et al. (2011). Rather, we focus on the typical steps for group-level DCM inference using PEB. Our experience with teaching SPM is that students appreciate having a concrete example, which they can then adjust to their own needs.

The steps below are also scripted in the '`code`' directory that you can download or clone from https://github.com/pranaysy/MultimodalDCM. This contains two sub-directories, one for fMRI and one for MEG, which themselves contain two groups of files: one of these are MATLAB files derived from SPM's 'batch' interface (filenames beginning with `batch*`), in which various analysis steps (batch 'modules') were created by the GUI, saved and then called from loops across subjects (all within the '`spm_master_script_dcm_*_peb_batch.m`'); the other consists of a script that implements exactly the same analyses, but with direct calls to underlying '`spm*.m`' functions, bypassing SPM's Batch interface (e.g., contained within the script in '`spm_master_script_dcm_*_peb_direct.m`').

2. The Multimodal Dataset

The dataset comes from a paradigm in which participants saw a series of faces and phase-scrambled faces, and made left-right symmetry judgments to each stimulus. There were 300 unique faces and 150 unique scrambled images. Half of the faces were famous and half non-famous, but we ignore this distinction in this tutorial. Each stimulus was presented for 900ms on average, followed by 2200ms on average. Each stimulus was repeated either immediately, or after 5-15 intervening stimuli, but again we ignore the effects of repetition here. Thus we only analyse two conditions: faces vs scrambled faces. See Lee et al. (2022) for DCM analysis of effects of repetition and recognition (familiar vs unfamiliar) in the fMRI data.

To give participants a break, the fMRI experiment was split into 9 runs, with approximately equal numbers of each condition per run, though to avoid delayed repetition across runs, a small number of these trials were dropped. In addition, in order to estimate the response versus inter-stimulus baseline,

six periods of 20s of a fixation cross were added after a block of 9-20 trials. For more details, see Wakeman & Henson (2015).

3. Background

3.1 DCM

DCM is a tool for inferring 'effective connectivity' between brain regions of interest (ROIs), based on explicit network models and assumptions about neural dynamics. DCM is a state-space model, consisting of 1) differential equations that relate changes in each latent neural variable to other variables in the network (depending on connections), and 2) an observation model that maps neural variables to the measured fMRI and M/EEG signals.

For fMRI, the neural model is based on simple exponential decay of activity within each area, offset by input from other areas, as captured by a simple bilinear approximation. For MEG, the neural model is much more complex and contains several differential equations with multiple parameters describing each ROI, based on knowledge of the neurophysiology of human cortical layers. For more background on DCM, see David et al. (2006), Moran et al. (2013), Pereira et al. (2021).

For fMRI, the observation model is a temporal model that maps brief changes in neural activity to the more dispersed BOLD impulse response (a so-called HaemoDynamic Model, HDM). For MEG, the observation model is a spatial model that maps certain neural variables to electrical/magnetic fields recorded by sensors outside the head.

3.2 Current Network (model)

Here we focus on 3 ROIs: left and right fusiform face areas (FFA) and bilateral early visual cortex (bVC). The right FFA is one of the peaks that survive correction for multiple comparisons in the group analysis of the contrast of faces versus scrambled faces (see Supplementary Figure A2.1 of Henson et al. (2019); https://www.frontiersin.org/articles/10.3389/fnins.2019.00300/full#supplementary-material); the left FFA appears if the cluster threshold is reduced from 30 to 10 or fewer voxels. We could define these two ROIs by saving each of them as thresholded clusters (as in Section 3.2 of Henson et al., 2019), but here, for fMRI, we just define them as spheres centred on significant peaks (see Section 4.3 below) For bVC, the centre of the sphere was just chosen manually to estimate midline early visual cortex.

We connect the three ROIs as shown in Figure 1. This assumes bidirectional connections between bVC and FFA, and bidirectional connections between hemispheres for FFA. Together with each ROI's (inhibitory) self-connection, these are the 'A' connections in DCM. We further assume that all of these fixed connections can be modulated by the presence of faces (vs scrambled faces). Finally, we will assume that the input (for all stimuli) enters bVC (but see Lee et al., 2022, for more nuanced treatment of possible inputs to the network).

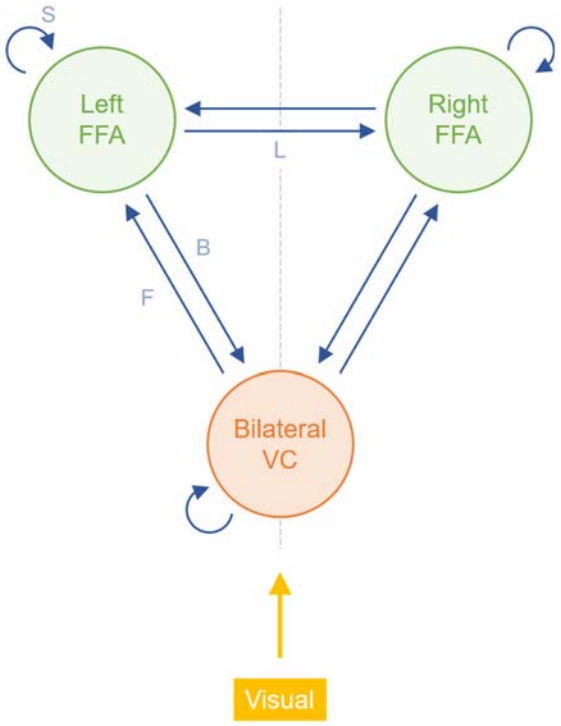

Figure 1. The 3-node DCM used for fMRI and MEG/EEG. F = forward, B = backward, S = self, L = lateral (note F, B and L have different properties in DCM for MEG/EEG only). VC = Visual Cortex; FFA = Fusiform Face Area.

DCM proceeds by comparing different models of the data through (an approximate lower bound on) the Bayesian model evidence, where models typically differ in their connections (parameters); normally those connections that are modulated by an experimental manipulation (here, by faces). When there are multiple subjects, one can create a single hierarchical model, enabling an Empirical Bayesian approach in which the mean and covariance of parameters across subjects can act as a prior on individual subject parameter values. Since DCM also assumes multivariate normal error terms (so-called 'parametric' assumptions), this approach is called Parametric Empirical Bayes (PEB); see papers by Zeidman and colleagues (Zeidman, Jafarian, Corbin, *et al.*, 2019; Zeidman, Jafarian, Seghier, *et al.*, 2019) for more details.

There are many models one could compare. For example, one could ask whether face processing modulates connections between ROIs, or whether it is sufficient to explain face-related responses in each ROI simply via modulations of each ROI's self-connections. If the latter 'self-only modulation' model were more parsimonious (had higher model evidence), then there would be no need to assume that faces change the effective connectivity between regions (and the traditional voxel-wise analysis of univariate statistics would be sufficient, as in Henson et al., 2019).

The tutorial consists of two main sections: 1) specifying and estimating a single-subject DCM for fMRI, and 2) estimating group-level DCMs for fMRI with inference based on model comparisons.

4. DCM for fMRI

Preprocessing of the fMRI data is described in Supplementary Material of Henson et al. (2019) https://www.frontiersin.org/articles/10.3389/fnins.2019.00300/full#supplementary-material, but if you do not want to repeat that, you can download the preprocessed image files and first-level models from the Figshare link below. More practical help on DCM for fMRI can be found in Chapter 36 of the SPM12 manual, https://www.fil.ion.ucl.ac.uk/spm/doc/spm12_manual.pdf.

4.1 Preparation

In order to run this tutorial, there are two preparatory steps involved: setting up the software environment, and organizing data. For preparing the environment, install SPM12 from https://www.fil.ion.ucl.ac.uk/spm/software/download/ somewhere on your local system. The results in this tutorial were obtained with release '`r7771`'. Once installed, clone or download the git repository at https://github.com/pranaysy/MultimodalDCM to a folder. This contains some small edits to SPM functions that are detailed in the README. Launch MATLAB and using the file browser panel, navigate to the cloned or downloaded repository folder. We will refer to this folder as '`base_dir`' in the tutorial as well as scripts.

Once the environment is ready, the data can be organized according to the level of pre-processing, with three possible starting points:

1. You could begin with raw data from OpenNeuro (https://openneuro.org/datasets/ds000117) and run the demo in the supplementary material of Henson et al. (2019) to preprocess the raw images into smoothed, normalised, slice-time corrected and realigned images. Note that the full data is about 85GB in size. Once you have downloaded and finished processing the data, you will have a BIDS directory tree with processed derivatives in the folder '`data/derivatives/SPM12`'. We refer to this folder as '`derpth`'.

   The data in this folder are not yet ready for fitting DCMs and need to be processed further. The SPM single-subject models will need to be reparametrized, their runs concatenated and re-fit to all the smoothed, normalised fMRI volumes, after which extraction of VOI time courses will be necessary. These steps are described in sections 4.2 and 4.3, with precise details given in the Appendix. DCMs can then be fit using the extracted VOI time courses and reparametrized SPM models.

If you are not starting with raw data from OpenNeuro and would like to proceed with pre-processed data, you have two choices:

2. You could download fMRI images that have already been preprocessed from Figshare in the file '`fMRI_ProcessedData_Individual_Runs.tar.xz`' (https://doi.org/10.6084/m9.figshare.25192793.v1) and extract into the '`data`' folder in '`base_dir`'. Note this will take almost 17GB after extraction. The data folder should now have a directory tree that looks like '`base_dir/data/derivatives/SPM12`'. We refer to this folder as '`derpth`'. Inside this folder there are 16 sub-directories called '`sub-01`', '`sub-02`' etc, each of which have a further sub-directory called '`fmri`', in which there should be three types of file for each of 9 runs: the 4D smoothed, normalised NIFTI images ('`swsub-*.nii`'), a MATLAB file with the trial definitions ('`sub-*_run-*_spmdef.mat`') and a text file with the 6 motion parameters from spatial realignment of the volumes ('`rp_sub-*_run-*.txt`'). Single-subject first-level models will need to be created using the same steps for reparametrizing, concatenating and VOI extraction as described in sections 4.2 and 4.3.

3. Lastly, you could start directly with processed DCM-ready data, which consist of extracted VOI time courses for the 3 ROIs: bVC, lFFA and rFFA, along with reparametrized SPM.mat files for each subject. If you use this dataset, you can skip sections 4.2 and 4.3 and proceed directly to section 4.4 for specifying DCMs. For this approach, download the file '`fMRI_DCMreadyData_VOI_TimeCourses.tar.xz` ' from the same Figshare link as above, and extract into the '`data`' folder in '`base_dir`'. The data folder should now have a directory tree that looks like '`base_dir/data/derivatives/SPM12`'. We refer to this folder as '`derpth`'. Note this will take up around 1.1GB of space after extraction. Each subject's data in consists

of one SPM.mat file and one file each for the three VOIs along with masks. These are the same files that would be produced by running the steps described in sections 4.3 and 4.3.

4.1.1 Initialisation

Open '`spm_master_dcm_fmri_peb_batched_script.m`' from the '`base_dir/code/fmri`' folder. There is some MATLAB code at the start that is necessary to start SPM and define some key variables. First start SPM12:

```
SPM12PATH = <insert path to your local install of SPM12>
addpath(SPM12PATH);
spm fmri
```

Then define some paths where you have downloaded the code and data from above:

```
base_dir = <path to where you have cloned the respository>
derpth = fullfile(base_dir,'data','derivatives','SPM12')
% above will exist if you have run preprocessing scripts for Henson et al,
% 2019, or else create these directories and download and extract
% "fmri_data.tar.gz" from Figshare as above
```

Next, add the '`code`' folder and all subfolders in it to MATLAB's path by running this line:

```
addpath(genpath(fullfile(base_dir, 'code')))
```

This ensures that all the code we provide is available in MATLAB's environment. This is necessary because our scripts for batch processing rely on functions in various files – some of which enable parallel processing, some are 'job files' used by SPM's batching interface while some are modified SPM functions, which have been updated for this tutorial. It is crucial to add this '`code`' folder to the MATLAB path after your local installation of SPM has been added to MATLAB's path and launched. This sequence of operations will put our '`code`' folder at the top of MATLAB's list of paths, which can be viewed by typing `pathtool` on the command line. Since the updated SPM functions we provide share the same filenames with the original SPM functions, adding the two in this exact order guarantees that when a function is to be executed, MATLAB will look for any version in our '`code`' folder first, since it is higher in the path. A list of modified SPM functions with a brief overview of changes is provided in the README file on the GitHub repository for this tutorial, linked earlier.

Then you can run the lines below to define some variables:

```
% If you have all raw data
% BIDS = spm_BIDS(rawpth);
% subs = spm_BIDS(BIDS,'subjects', 'task','facerecognition');
% runs = spm_BIDS(BIDS,'runs', 'modality','func', 'type','bold',
'task','facerecognition');

% Else specify subjects and runs manually
Subs   = compose('%02g', [1:16]);
nsub   = numel(subs);
subdir = cellfun(@(s) ['sub-' s], subs, 'UniformOutput',false);
```

Then, if you have access to multiple cores (parallel processing in Matlab), you can run below (if not, set "numworkers" to 0):

```
numworkers = nsub; % Number of workers for distributed computing
if numworkers > 0
```

```
        delete(gcp('nocreate')) % Shut down any existing pool
        parpool(numworkers);
    end
```

4.2 Re-parametrising each subject's fMRI model

If you have downloaded the data and models from Henson et al. (2019), two further preprocessing steps are required before we can create DCM models: 1) re-parametrising the single-subject (1st-level) fMRI general linear model (GLM) for each subject, and 2) extracting the (adjusted) fMRI timeseries for each ROI and each subject.

The first step of re-parametrising the GLM involves defining two conditions: 1) all stimuli (faces and scrambled), to be used as the driving input for DCM ('C' matrix) and 2) faces only (collapsing famous and unfamiliar), to be used as a modulatory input for DCM ('B' matrix). It also involves concatenating the 9 runs into a single run for convenience. These steps are described in Appendix 6.1, where they can either be implemented by running some MATLAB code on the SPM.mat files produced by the Supplementary Section 2 of Henson et al. (2019), or by re-creating the SPM.mat files from scratch using SPM's batch interface.

4.3 Defining ROI (VOI) timeseries

DCM fits fMRI timeseries for a number of ROIs. Those timeseries are extracted by taking the first temporal mode[1] across a number of voxels within a specified volume of interest (VOI) that defines the ROI. In SPM, a VOI can be defined by one or more constraints. The full options are described in chapter 36 (section 36.3.3) of the SPM manual (see also (Zeidman, Jafarian, Corbin, *et al.*, 2019)). Here we will define our VOI by two constraints: 1) voxels within a sphere of 10mm radius from MNI coordinates that we will provide, 2) voxels that showed an increased response at the onset of stimuli (relative to interstimulus baseline), as defined by a T-contrast thresholded at $p<0.001$ uncorrected. The second constraint ensures that voxels within the sphere are likely to be gray-matter that is generally responsive to the stimuli (i.e, removing voxels within the sphere that are white-matter or CSF). Of course, further constraints could be added (e.g, a gray-matter mask), or other definitions used instead (e.g, a mask image created by thresholding each individual subject's contrast of faces versus scrambled), but they are not explored here.

SPM produces a `VOI*.mat` file for each ROI and each subject (e.g, '`sub-01/fmri/VOI_lFFA_1.mat`' for the lFFA of subject 1), which are passed to DCM below. The precise steps needed to create these files are described in Appendix 6.2.

4.4 DCM definition: for single subject

Unfortunately DCM definition is not yet batched in SPM, so we need to go through SPM's GUI by pressing the 'DCM' button (the following mirrors the PDF available here: https://github.com/pzeidman/dcm-peb-example).

In MATLAB, use the file selector on the left hand side to change the working directory to sub-15. We now walk through each step involved in specifying the 'full' DCM model, which we will subsequently estimate.

• Click the big Dynamic Causal Modelling button in main SPM window. In the grey window that appears, click '`Action`' and then '`specify`'.

• In the file selector, click the SPM.mat file from sub-15 on the right hand side and then click 'Done'. This provides DCM with the timing of the experimental conditions.

[1] This is from a singular-value decomposition of the data to extract the first temporal and spatial singular vectors. If all the voxels were equivalent, the temporal mode would be equivalent to the average over all voxels.

• You'll be asked for a name for the new DCM. Type: '`Full`' and press enter.

• You'll be asked to select the VOIs (volumes of interest) for this subject. These are the timeseries for each brain region, which have already been prepared. The order is important - the same order will be used for the regions in the DCM. For consistency with this tutorial, select in order: bVC, lFFA, rFFA and click '`Done`'.

• Now you are asked which experimental conditions to include – include both '`All`' (faces+scrambled) and '`Faces`'; select '`yes`' for both.

• For VOI timings, keep `1` (second), ie half TR, since the data were temporally interpolated to match the acquisition time of the middle slice.

• For Echo Time (TE), change to `0.03` (30ms), which was used on this 3T scanner.

• You are now asked to set certain options for the model. Keep defaults of `bilinear`, `one` state per region, `no` stochastic effects, `no` centred input, and fitting fMRI `timeseries`.

• You are now asked which connectivity parameters you want switched on (free to be informed by the data) and which you want switched off (fixed at their prior expectation of zero). These are connectivity parameters in matrix A from the DCM neural model, which is the average connectivity over experimental conditions. The self-connections (on the leading diagonal) are always switched on (and are always negative, to ensure the network dynamics converge to a steady-state), so this screen is really asking you to select which between-region connections to include. Each column is an outgoing connection and each row is an incoming connection. Switch all these between-region connections on, according to Figure 2 and then press done. (Tip: holding your mouse pointer over a button will identify the nature of the connection.)

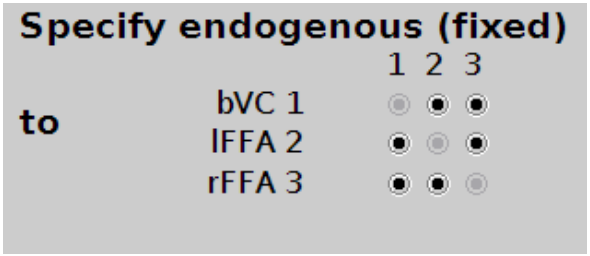

Figure 2: DCM specification for matrix A.

• Next you are asked about each of the experimental effects, starting with '`All`' stimuli. The buttons on the left are the driving inputs (matrix C). Set all stimuli to drive bVC only, as in Figure 3 and press done.

Figure 3: DCM specification for driving inputs (C) for all stimuli.

• Next you are asked about the effect of '`Faces`' stimuli. For these trials, we will use the buttons on the right, which reflect the modulatory inputs (matrix B), which increase or decrease the strength of particular connections as a function of whether or not the stimulus on each trial was a face. Here, we allow this experimental condition to modulate all B connections, as shown in Figure 4.

Figure 4: DCM specification for modulatory inputs (B) for faces only (in full model).

You will receive a polite '`Thank you`' and a file called '`DCM_Full.mat`' will have been created in this subject's directory.

4.5 Estimating full model for single subject

Before running this step, copy '`DCM_Full.mat`' to a new file '`DCM_Full_sub-15.mat`' in the same sub-15 directory. This is because this way of estimating of a DCM will update that file with the posterior estimates, yet we do not want to use these posteriors when we re-fit all subjects later in this demo. To fit DCM to the data, you can press main DCM button, select '`Action:… estimate (time-series)`' and select the '`DCM_Full_sub-15.mat`' file. Or equivalently (to avoid copying step), you could run this code:

```
load(fullfile(outdir,'DCM_Full.mat'));
DCM = spm_dcm_estimate(DCM);
save(fullfile(outdir,'DCM_Full_sub-15.mat'),'DCM');
```

Once estimated, press the main DCM button again, select '`review`' and choose the '`DCM_Full_sub-15.mat`' file. Select the option '`effects of All`' under '`review`', and you should see Figure 5. All stimuli (faces and scrambled) functioned as the driving input (C), which we specified to bVC, and the results show that this input was needed (posterior probability close to 1), as shown in top right panel of bottom section (the top left panel of that section shows actual values, in Hz). There are no results for the B connections, because we only allowed the second input (Faces) to modulate connections.

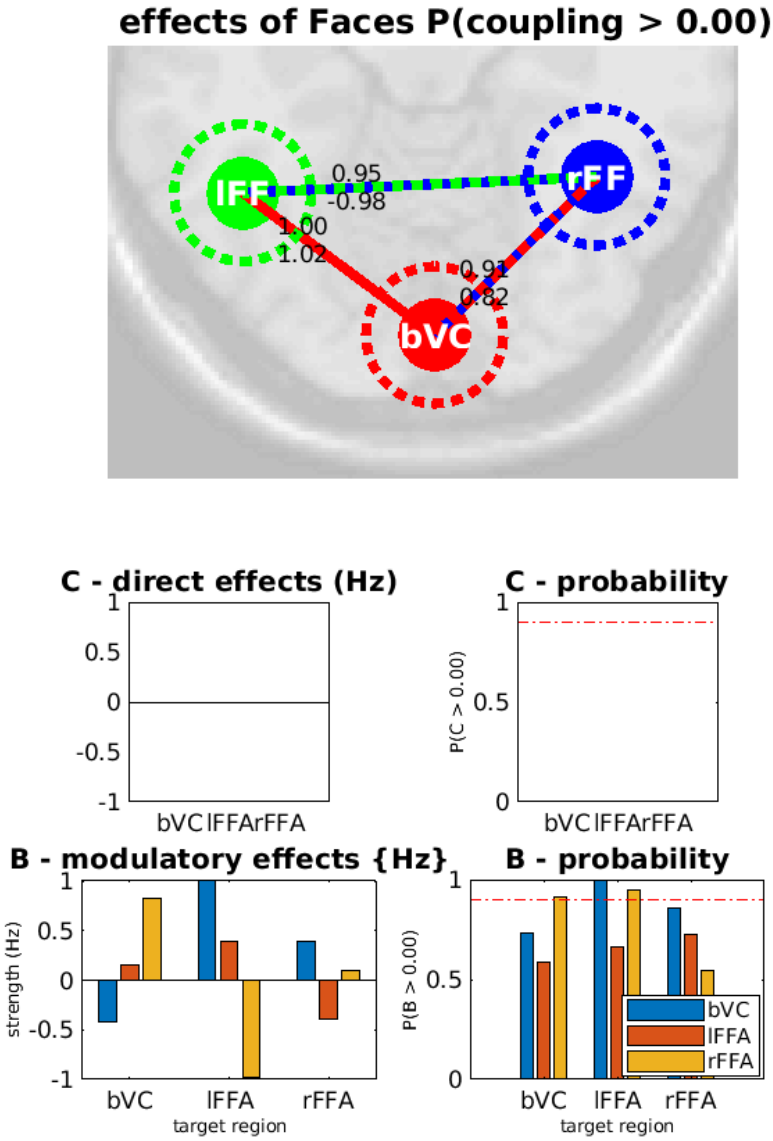

Figure 5. Single-subject fit of full model – Effects of All (Faces + Scrambled)

If you go back to '`review`' window, and select '`effect of Faces`' instead, you will see Figure 6. In the bar graphs for the modulatory (B) connections (bottom right), you can see that some are needed (above the red dotted line for 0.9 probability), eg forward connection from bVC to lFFA (middle blue bar). Again, their values are shown in bottom left panel.

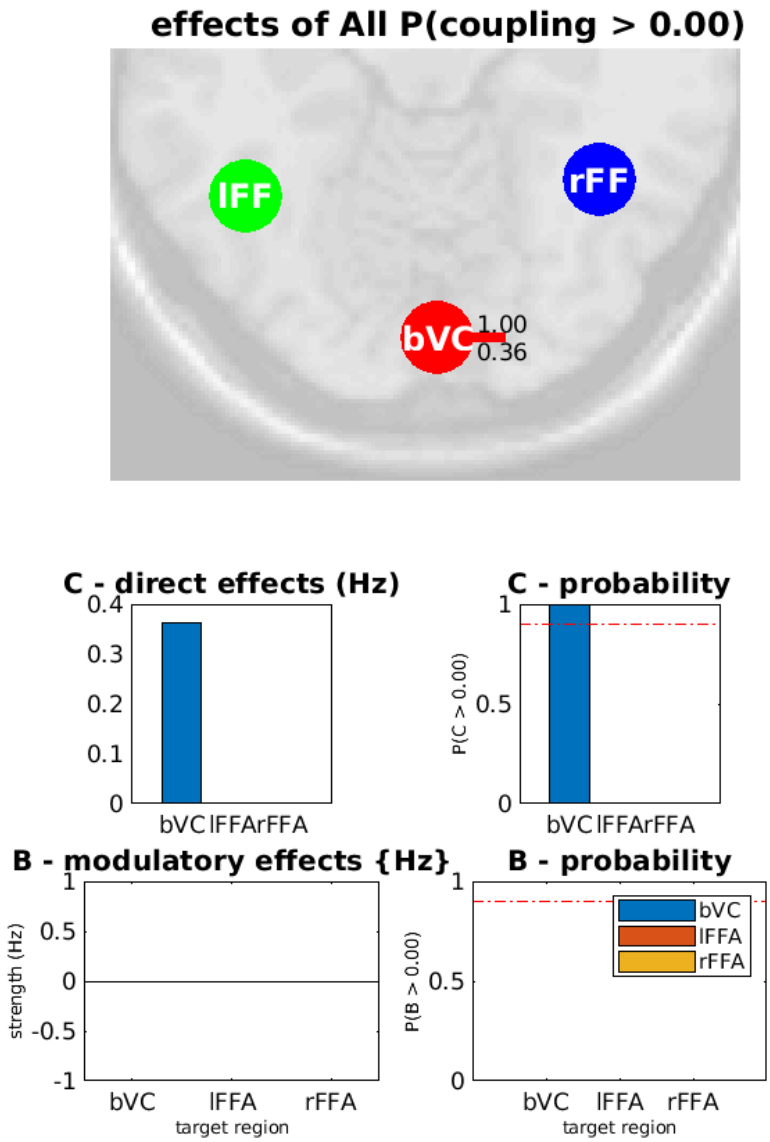

Figure 6. Single-subject fit of full model – Effects of Faces

You can review the '`fixed (A) connections`', but these are not very interesting in this context. You can also specify '`contrasts of connections`' (parameters), which we will not explore here (we will keep our inference at group level, across models; see later). You can also review '`location of regions`', and the '`inputs`' (from the SPM.mat) file, but more interesting are the '`outputs`', shown in Figure 7. The solid blue line shows how well DCM fits the data (dotted line) in each ROI. Finally, you can also examine the '`kernels`' if you are interested in the neuronal and haemodynamic parameters for each region (which we normally are not).

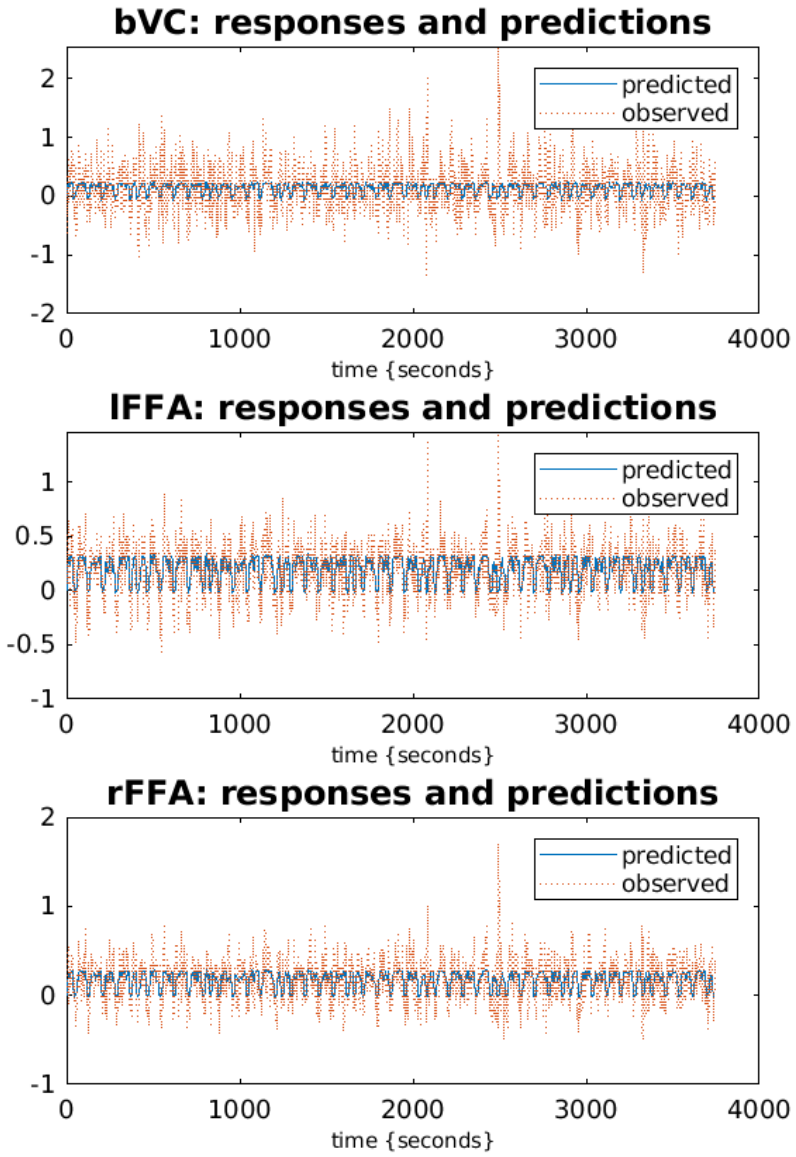

Figure 7. Single-subject fit of full model – Outputs

Make sure to quit the review window when finished.

4.6 Group DCM

4.6.1 GCM definition: Replicating across subjects

Having specified a 'full' DCM for a single subject, we can now use this as a template to specify the same DCM for every subject, and enter into a 'Group Causal Model' (GCM). All subjects' models will be the same, except the timeseries and timing of the experimental input, which will be customized for each subject. The batch GUI can replicate an existing specified model across subjects, filling in each subject's data in specific fields. This approach works when all subjects have exactly same timings, but is not currently designed to accommodate differences in timings that may result due to variable number of scans in some runs. In our data, subject 10 had a different number of scans for a run, and therefore input timings in this subject's model need to be specify separately. You can proceed in one of two ways:

A. Use the batch GUI to replicate the template model over all subjects except sub-10 to obtain a GCM of 15 subjects. Then separately specify sub-10's model and 'wedge' it in the GCM.

B. Use a helper script called '`GCM_Helper.m`' in '`base_dir/code/fmri`' which automates the creation of GCM from a template model and each participant's data, and handles differences in timings due to variable number of scans. This directly creates a GCM with all 16 subjects.

Here we illustrate option A using the batch GUI:

Open the batch GUI. From the menu at the top of the Batch Editor, click SPM → DCM → DCM specification → DCM for fMRI → Specify group:

• Output directory - double click Output directory to bring up the file selector. We will store the GCM file in the templates folder located at '`base_dir/fits/batch_gui/fmri/templates/GCMs/Full`'. Navigate to this directory and pressing the single dot '`.`' (which indicates current directory). Then press Done.

• Output name - double click, then type '`Full`' and press OK.

• Full DCM - double click, then in the file selector, select the full DCM we made earlier for sub-15. It is named '`DCM_Full.mat`'. Select this on the right hand side, then press Done.

• Alternative DCMs – leave empty.

• SPM.mat files - We will now select all subjects' SPM.mat files except sub-10's, which contain the timing information for each subject. Once in the base DCM directory, you should now see a list of all subject directories. Press the small '`Rec`' button (for 'recursive search'). This will search through all subject directories and pick out their SPM.mat files (since the '`filter`' is set for SPM.mat files only). Check that all 16 are selected at the bottom of the file selector window, remove the file corresponding to sub-10 by clicking on it and press '`Done`'.

• Regions of interest - Now we'll select the timeseries (VOIs) for each subject except subject-10. Click Regions of interest then click New: Region (VOI files). Do this three times, so you get three entries in the batch that say '`Region (VOI files)`'. From the base DCM directory, edit the filter on the selection box from '`^VOI_.*\mat$`' to add '`bVC`', i.e. to make '`^VOI_bVC.*\mat$`' (this is a 'regular expression' in linux that matches certain strings, here in the filenames). Then click '`Rec`', which will search through all the subjects' folders selecting the 16 bVC VOI files. Again, remove the files corresponding to sub-10 and press '`Done`'.

• Repeat for the other two ROIs, just changing the filter name, but importantly the order you enter them must be lFFA and then rFFA (to match order in DCM specification files) and make sure you deselect files for sub-10.

• Click File → Save batch and save it in '`saved_from_batch_interface`' sub-directory in '`base_dir/code/fmri`' as '`batch_dcm_create_gcm`' (just for record, eg to check in case any manual mistakes made).

Then press the green play button to run this module.[2] You'll see one DCM file in each subject's folder named '`DCM_Full_m0001.mat`' (the full model). Their filenames will also be collated into a single cell array and saved in a mat file named '`GCM_Full.mat`' in the templates directory. This file consists of paths to each subject's specified DCM files. Load it into the MATLAB workspace by running:

```
model = load('GCM_Full.mat');
GCM = model.GCM;
```

[2] In the accompanying tutorial on MEG, we will demonstrate the use of 'dependencies' within SPM's batch interface, where the output(s) from one module can be specified as the input(s) to subsequent modules, even though those output files have not yet been created, in order to create a single 'pipeline' of modules before running it.

Next, specify a model for sub-10 by following the steps in section 4.4, and entering sub-10's SPM.mat and VOI files instead of sub-15's. After saving this file, close the DCM GUI, navigate to sub-10's data directory, find the 'DCM_Full.mat' file and assign its full path to a variable in MATLAB's workspace:

```
DCM10 = '/home/.../base_dir/data/derivatives/.../sub-10/CatGLM/DCM_Full.mat';
```

Now, insert path to sub-10's specified model in the 10th index of the GCM cell array by running:

```
GCM = {GCM{1:9}, DCM10, GCM{10:end}}';  % Make sure this is 16×1, not 1×16
```

Lastly, save this GCM as 'GCM_Full.mat' and overwrite the existing file.

4.6.2 Estimating GCMs

Having specified DCMs for every subject, we now fit them to the data. In the Batch editor, click File → New Batch. Then click SPM → DCM → DCM estimation. Fill out the batch as follows:

• Select GCM *.mat - double click, then select the file named 'GCM_Full' we created earlier from the templates directory. Then press Done. The file we selected contains a cell array of DCM filenames.

• Output - single click Output, then click 'Create group GCM *.mat file' and for output, select the 'base_dir/fits/batch_gui/fmri' directory and name it 'Full'. For Estimation type, choose 'Full + BMR (default)'; other items can be left on their defaults too.

Save this batch by clicking File, Save Batch as 'batch_dcm_fit_gcm' in 'saved_from_batch_interface' sub-directory in 'base_dir/code/fmri' for record, and press green play button. This will take a long time, though if you have multiple cores, you can speed up DCM estimation by using MATLAB's parallel computing: this 'use_parfor' switch has been on in the modified version of spm_dcm_fit.m in the 'code' directory.

This will fit each full model to the subject's data independently. The results will overwrite the DCM files in each subject's folder.

4.6.3 Diagnostics

Having completed the estimation of the first-level DCMs, it is a good time to perform some diagnostics on the models. First, in the main Matlab window, change to the analyses directory ('base_dir/fits/batch_gui/fmri') and load the 'GCM_Full.mat' file and then execute following SPM command within Matlab console: 'spm_dcm_fmri_check(GCM)'.

The output of this command should look like Figure 8.[3] This is a graphical representation of the GCM file, where the long coloured bar indicates the explained variance of the DCM for each subject, and the columns correspond to models per subject, in our case just the one 'Full' model. This column will have the explained variance calculated (averaged across all ROIs). Here we clicked on the model for subject 2, who had explained variance 39.16%. (Depending on your SPM version, you may get slightly different explained variance.) The lowest of 11.78% is for subject 12, while the highest is 49.60% for subject 14. As explained in DCM literature, % variance explained is not the ideal criterion (model evidence captures this accuracy, but also 'adjusts' for model complexity), but as a rule-of-thumb, one might be concerned if DCM explained <10% of the variance (e.g, there could be excessive noise in data from one participant, or incorrect GLM specification, or just poor DCM choice)

After clicking on a participant's full model (left hand column of the figure), you can click the Diagnostics button, to further explore, eg, the predicted timeseries, as we did for the single subject DCM fit for subject 15 in Section 4.5.

[3] Press "CTRL-" or "CTRL+" on "SPM Figure" menu if font too big/small.

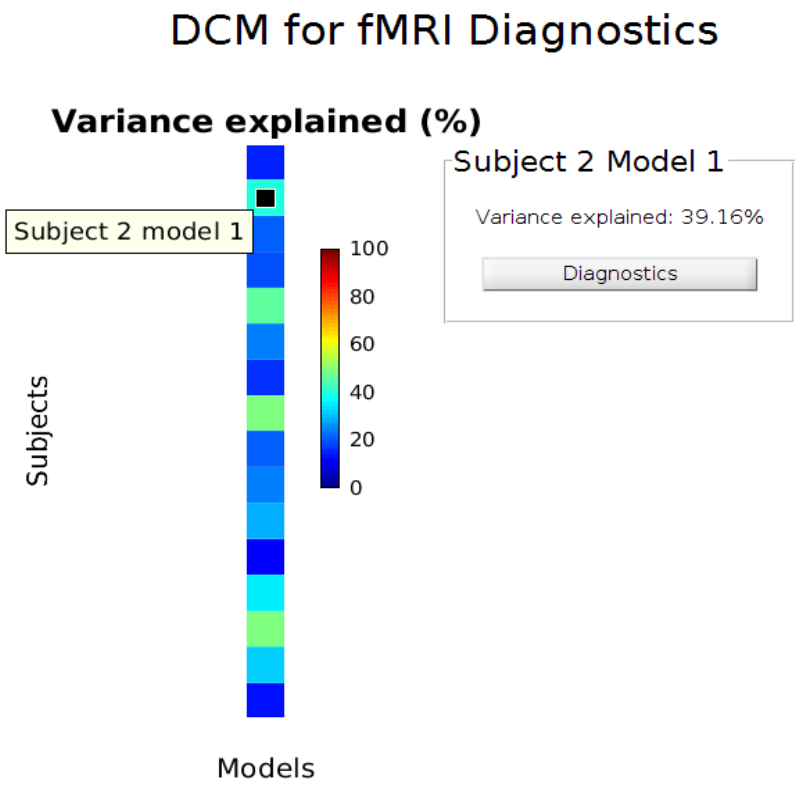

Figure 8. Diagnostics of fits of full model for each participant

4.7 Second level analysis: PEB

Having estimated each subject independently in the previous section, we now want to get a single measure of model evidence at the group-level (hierarchical) model. We might also want to examine differences between subjects by specifying group covariates like age or sex (see later for an example), though for now we just estimate the simple group average, for simplicity.

4.7.1 PEB model specification

Go to the main SPM window and click `Batch`. Then click `SPM` → `DCM` → `Second level` → `Specify / Estimate PEB`.

• Name - This is a name for the analysis, which you can enter '`Full`', since SPM with prepend '`PEB`' to create file '`PEB_Full.mat`'.

• DCMs - Double click, then navigate to the '`base_dir/fits/batch_gui/fmri`' folder and select the file named '`GCM_Full.mat`'.

• Selected DCM index - Leave this on the default value of `1`.

• Covariates – keep as '`none`', but later below, we will add age here.

• Fields - We are only going to take parameters from DCM matrix B to the group level. Click Fields, then click '`Enter manually`'. Double click '`Enter manually`' and type: `{'B'}` including the curly braces, then press OK.

• Leave rest as default

Now save the batch by clicking `File`, `Save Batch` and give it a name like '`batch_dcm_fit_peb`'. Then run the batch by pressing the green play button. This will create and estimate the group-level PEB model and store the results in a file called '`PEB_Full.mat`'.

4.7.2 Model comparison: automatic search

To make use of the PEB model, we need to perform a model comparison. The simplest form of model comparison to run is an automatic search, which will prune parameters from the PEB model that do not contribute to the model evidence. The software will specify and compare hundreds of candidate reduced PEB models, in which different combinations of parameters have been switched off. This search can be performed quickly owing to a method called Bayesian Model Reduction (BMR), in which the parameters and model evidence for any nested model can be estimated from the full model fit by simple equations, without needing to re-fit each nested model to the data. Moreover, we can also average the parameters (connection strengths) across the whole model space, weighted by the model evidence for each model; a process called Bayesian Model Averaging (BMA).

In the batch editor, click `File` → `New Batch` then click `SPM` → `DCM` → `Second level` → `Search nested PEB models`. Fill it out as follows:

• Select PEB file – Select '`PEB_Full.mat`' created above

• DCMs - Select the file '`GCM_Full.mat`' – this file is only needed because it contains information about the full model needed for graphical output.

• Null prior variance - This determines the null hypothesis for each connectivity parameter - i.e. what prior variance constitutes a connection being 'switched off'. Set this to `0` (zero).

Save the batch as '`batch_dcm_peb_searchbmr`' and press the green play button. This will produce a file called '`BMA_search_PEB_Full.mat`', as well as three windows:

The window titled '`BMR - all`' (Figure 9) details 128 candidate PEB models from the final iteration of the automatic search (this number of 2^7 is because 7 out of total 9 'B' matrix parameters were identified by the procedure to produce the least reduction in model evidence when switched off individually). The top left plot shows the log model evidence for each PEB model and the top right shows these values converted to posterior probabilities. Note that typically no single model wins (in sense of conventional probability >0.95), though a few models are much more likely than remaining ones – for instance, here model 128 has a moderately high probability (~0.85).

The second row shows the parameters of the PEB model before the search (left) and after the search (right). Only two parameters have been pruned away because they did not contribute to the model evidence (free energy) – the fourth and sixth. We will return to the identity of these parameters shortly.

The bottom left plot shows the parameters that were switched on (white) and switched off (black) in each model from the final iteration of the search. For example, B(1,1,2) – the modulatory effect of faces on self-connection of bVC – was switched off in the first 64 models and switched on in the second 64 models. Finally, the bottom right plot shows the posterior probability for each PEB parameter. This is computed by comparing the evidence for all models (out of the final 128) that had the corresponding parameter switched on, versus all models that had that parameter switched off.

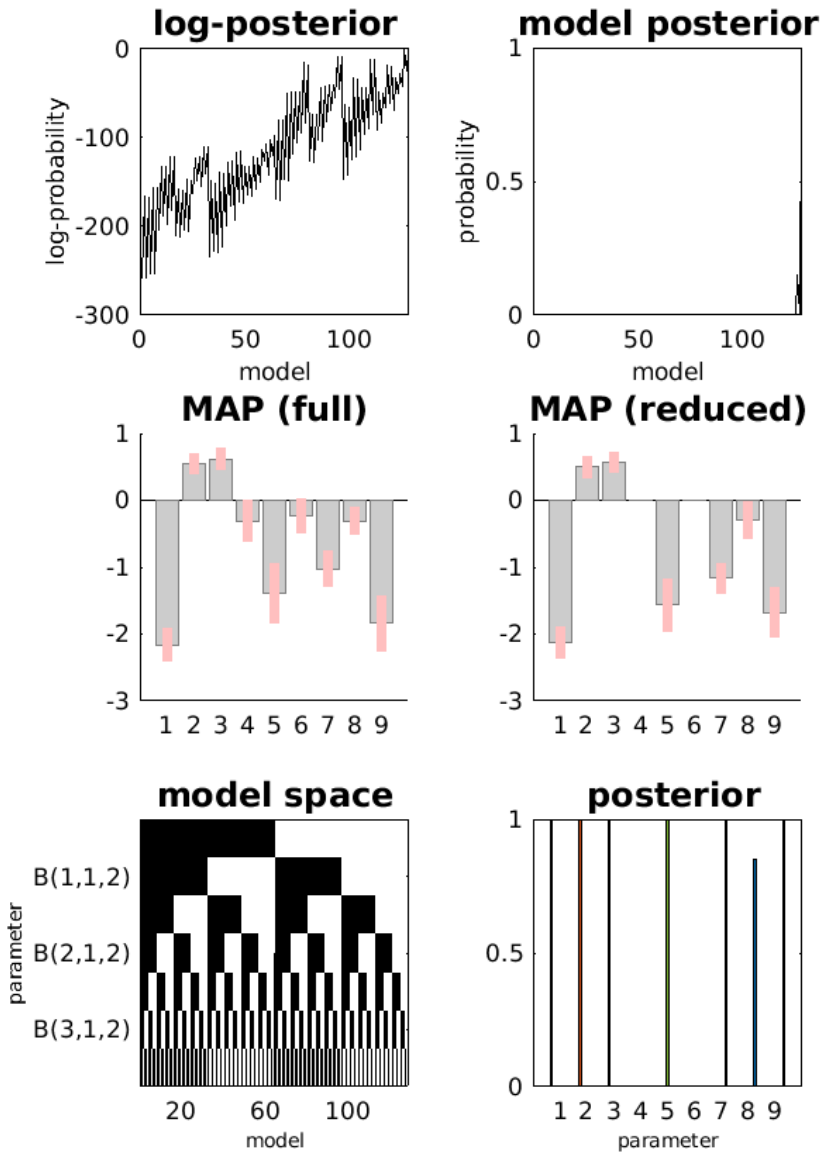

Figure 9. PEB BMR automatic search of whole model space: "BMR-all" window

With a large model comparison such as this, it is unusual to find one model that is the overall winner. Instead of considering the individual models, it is generally more informative to consider the BMA - the weighted average of the parameters over models. The window titled BMC (Bayesian Model Comparison, Figure 10) shows this average, with plots organised into three rows. The top row shows parameters from the estimated PEB, while the middle row shows parameters from the BMA, with their respective posterior probabilities in the bottom row. The bottom row shows that all parameters have posterior probabilities close to 1 except three – the eighth parameter has a probability of ~0.85, while the fourth and sixth parameters, which were pruned away. This suggests that parameters 1, 2, 3, 5, 7 and 9 are needed to explain the effects of faces over scrambled faces.

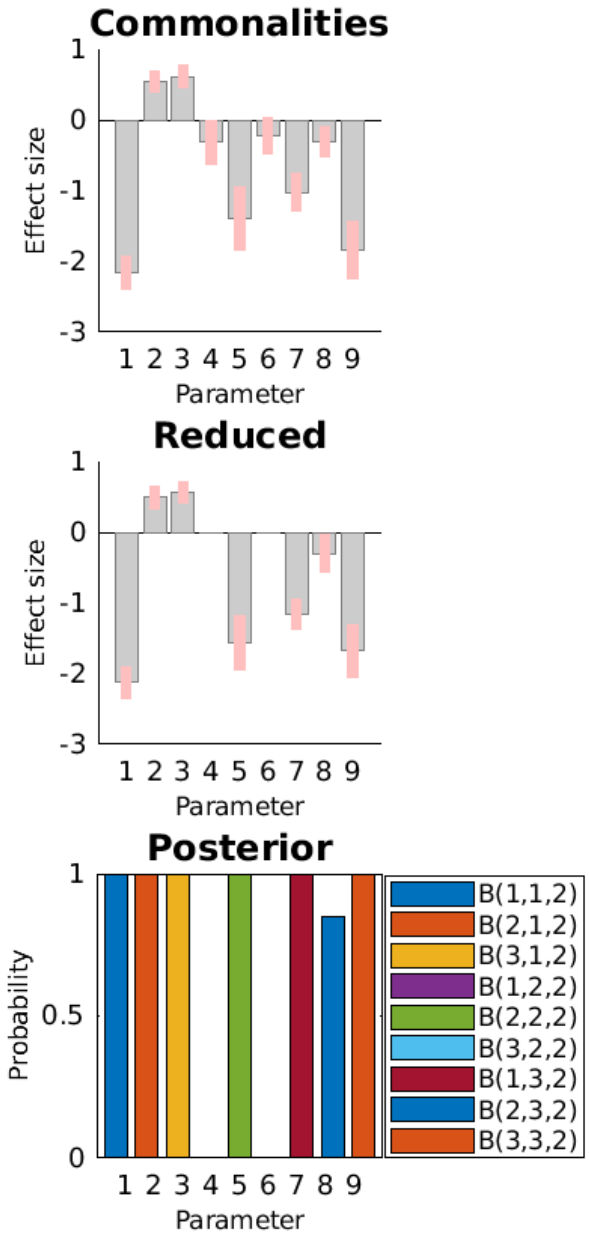

Figure 10 PEB BMR automatic search of whole model space: "BMC" window

The final window, titled '`PEB - Review Parameters`' (Figure 11), is an interactive tool that provides an easier way to explore the results (should you need to open this tool yourself at a later stage, then the command is: '`spm_dcm_peb_review(BMA, GCM)`', after loading the BMA, PEB and GCM files into Matlab).

• The boxes in top left give the number of regressors (covariates – just single group mean here) in the between-subjects design matrix, the number of DCM parameters (9 'B' parameters passed to PEB) and the number of subjects (16). The between-subjects design matrix is shown below, with one subject per row and one covariate of mean across subjects.

• The estimated between-subject covariance matrix. The diagonal is the estimated between-subjects variance for each parameter, where the more white they are, the greater the between-subjects variability.

Clicking on each item identifies the corresponding parameter (e.g, the backward connection from lFFA to bVC is most variable across subjects).

• The parameters are grouped by covariate, but here only one, so select '`Commonalities`'.

• Optionally, the parameters can be thresholded to just focus on the most probable effects. The first drop-down menu switches between thresholding based on the free energy (model comparisons with/without each parameter) and thresholding based on the posterior variance (the pink error bars). Where possible, we recommend selecting free energy, to accommodate interactions between parameters. The menu on the right is used to select the threshold, eg >0.95.

• The bars are the parameters relating to the selected covariate. Pink error bars are 90% credible intervals. Clicking on a bar shows the name of the parameter, its expected value, and its probability calculated using the option selected above.

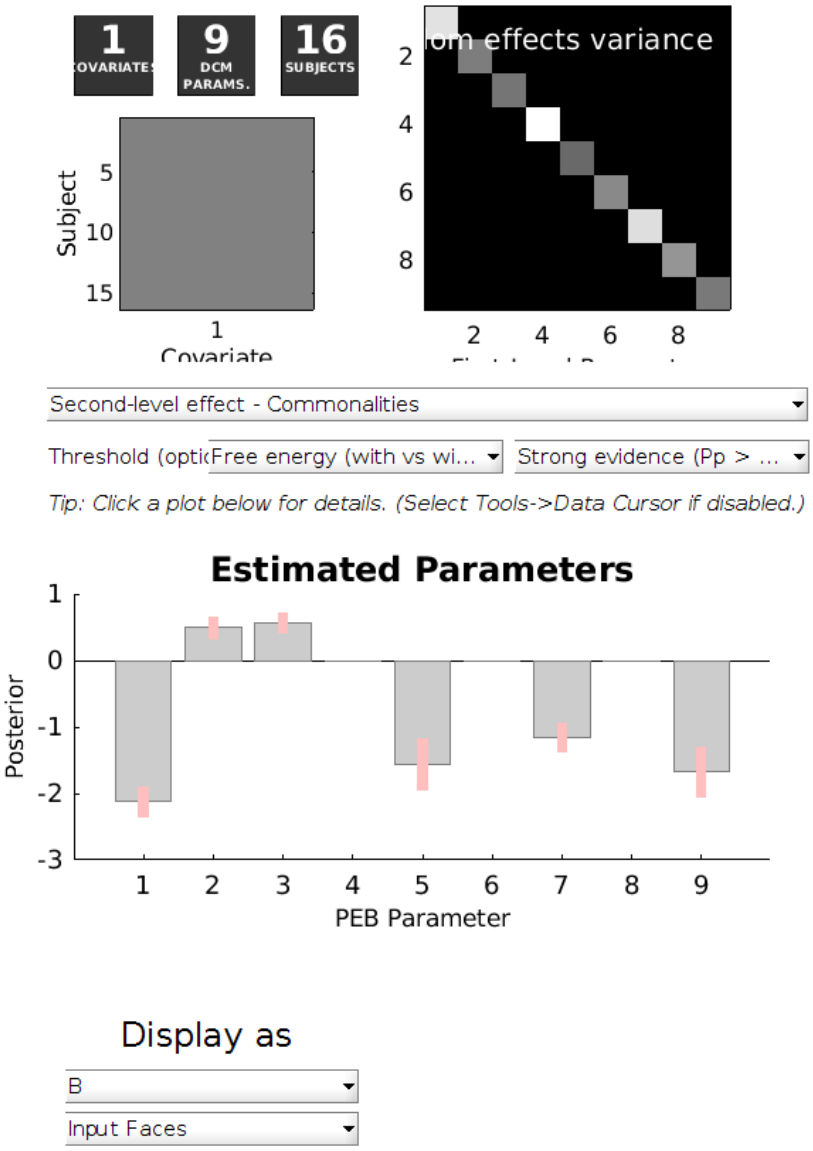

Figure 11 PEB BMR automatic search of whole model space: "PEB - Review Parameters" window

• Use the menus at the bottom of the window to display the parameters as a connectivity matrix. This is the same information as displayed in the bar chart, but shown in the same format as the connectivity matrices in DCM. Each column is an outgoing connection and each row is an incoming connection. First use the selector higher up in the window to choose the '`Second-level effect`' of Covariate 1 (which is simply the mean across participants). Then click '`Please select a field`' and choose the parameters of interest - e.g. '`B`' for the modulatory inputs (matrix B of the DCM neural model). A small window will pop up to select the input (modulator), which defaults to the first input (all faces and scrambled), which we did not allow to modulate, so we need to change to '`Input Faces`', which will update the popup window with the connectivity matrix to match Figure 12. This shows posterior probability for each connection, coloured by positive or negative sign.

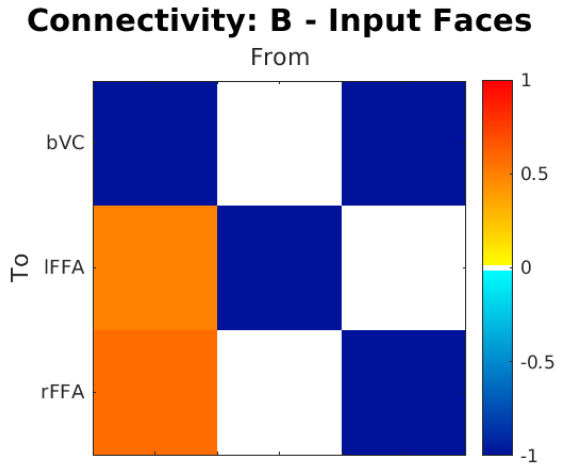

Figure 12 PEB BMR automatic search of whole model space: Thresholded B connectivity matrix

One can see that all self-connections (on leading diagonal) are needed, and are negative for all three nodes. A negative modulation means less self-inhibition when faces (since the self-connections in the A matrix are always constrained to be negative), which will tend to result in greater activity for that region (though this also depends on modulations of afferent connections from other regions). This is because, in DCM, the total self-connection strength is -0.5 * exp(A_ii + B_ii) Hz, so if B_ii decreases with faces, the connection strength because less negative.

One between-region connection from rFFA to bVC (backward) is negatively modulated, while both forward connections from bVC to lFFA and to rFFA are positively modulated during face processing.

4.7.3 Model comparison: families

The automatic search of all possible reducible models from the full model may not be sufficient to answer your questions. While it can return BMA estimates of connections that are needed, one might have a more general question that spans more than one connection, e.g, 'Do we need (modulation of) any backward connections from left/right FFAs to bVC?', or 'Do we need any lateral connections between hemispheres?'. First, we are going to ask the question 'Do we need any modulations of connections between ROIs, beyond modulations of self-connections?' – i.e., use DCM to ask whether there is any evidence of effective connectivity during face processing (that cannot simply be explained by local activation). In order to do this, we test whether one or more combinations of between-region connections are modulated by Faces. This involves specifying multiple 'nested' models with different combinations of between-region connections being modulated, i.e. switched 'off' or 'on' in the B-

matrix. The model space consisting of such 'nested' versions of the full model, which features modulation of all between-region connections, can then be divided into different 'families' (partitions) according to which types of modulation are enabled. Below, we will distinguish models according to whether they contain self-modulations, modulations of 'forward' connections (from bVC to FFA), modulations of 'backward' connections (from FFA to bVC) or modulations of 'lateral' connections (between lFFA and rFFA).[4] With these four types of connections, we can therefore define $2^4 = 16$ models in total:

1. Modulation of forward, backward, lateral and self-connections
2. Modulation of forward, backward and lateral but not self-connections
3. Modulation of forward, backward and self-connections but not lateral ones
4. Modulation of forward and backward connections only
5. Modulation of forward, lateral and self-connections but not backward ones
6. Modulation of forward and lateral connections only
7. Modulation of forward and self-connections only
8. Modulation of forward connections only
9. Modulation of backward, lateral and self-connections but not forward ones
10. Modulation of backward and lateral connections only
11. Modulation of backward and self-connections only
12. Modulation of backward connections only
13. Modulation of lateral and self-connections only
14. Modulation of lateral connections only
15. Modulation of self-connections only
16. No modulation of any group of connections

By comparing the family of models 1–14 that include forward and/or backward and/or lateral modulations with the family of models 15 and 16 that do not contain forward, backward and lateral modulations (only with or without self-modulations), we can test a similar hypothesis to the previous section i.e., whether between-region connections are needed. The subtle difference is that the precise question now being asked is no longer whether all between-region connections are needed, but whether at least one type of between-region connection (forward and/or backward and/or lateral) is needed, and furthermore, if this holds regardless of whether or not self-connections are also modulated.

4.7.3.1 Family-BMC for between-region connections

The first step involves creation of the sixteen template models for each model in the list above, which correspond to the matrices shown in Figure 13.

[4] Note we are always combining across hemispheres, though one could of course expand the model space to ask whether forward, backward or self-connections are needed in specific hemispheres. However, if one does not care about hemispheric differences, we are reducing the problem of model dilution by not considering models that differ in modulations between hemispheres.

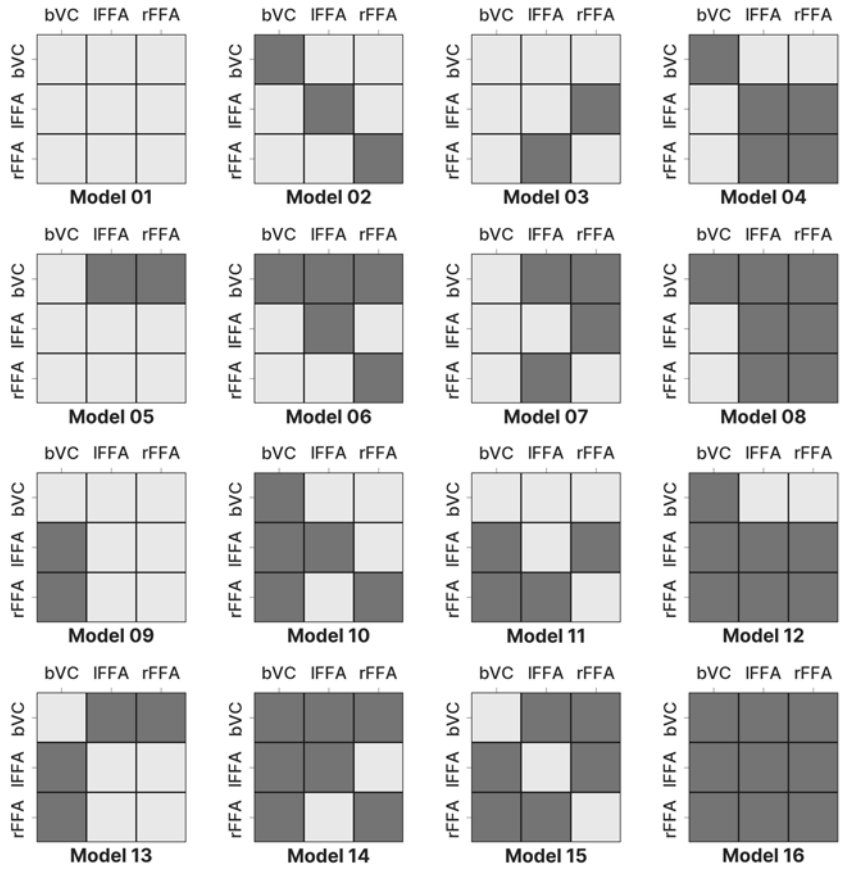

Figure 13: Model space for family-wise comparisons. White squares are connections that can be modulated (have value 1), whereas grey squares are connections that have been turned off and cannot be modulated (they have value 0).

This can be done through the GUI by loading the full model (for one subject), switching corresponding B-matrix connections off, and saving as a new file in the directory '`base_dir/fits/batch_gui/fmri/templates/GCMs/families`', as done for the '`Self`' model in Section 4.7.3. Or this can also be done more efficiently using simple MATLAB functions through the scripting interface, as illustrated on lines 766-819 in the script `spm_master_script_dcm_fmri_peb_batch.m`.

Either way, the resulting DCM definition files needed to be loaded into a GCM cell array, which now contains only one row, but sixteen columns, each column corresponding to one of the models (in same order as above). Note that because we will re-use the PEB that we estimated in the previous section, which already contains the full model for all subjects, we do not need to specify the alternative models for every subject; SPM will realise that the GCM now just contains the model space (for one subject), which is sufficient to use BMR to estimate all the nested (alternative) models for all subjects.

Once you have the GCM cell array in the MATLAB workspace, load the PEB model we fit earlier, and reduce the model space using direct calls to the following underlying SPM functions:

```
load('fits/batch_gui/fmri/PEB_Full.mat');

[BMA, BMR] = spm_dcm_peb_bmc(PEB, GCM);
```

Running this will show a window with BMC for all 16 models, along with parameter estimates as shown in Figure 14. The model space in the top-left panel reflects the 16 models we defined earlier. Based on

the middle left panel, model 1 with modulation of all specified connections in the B-matrix appears to have moderately high evidence (~76%), while model 3 with modulation of all connections except lateral ones has some evidence (~24%). This is also reflected in the parameter estimates in the BMA, where estimates of all parameters have a posterior probability of ~100% except parameters 6 and 8, each of which have a probability of ~76%. Next, we proceed to demonstrate partitioning of the model space into families for inference.

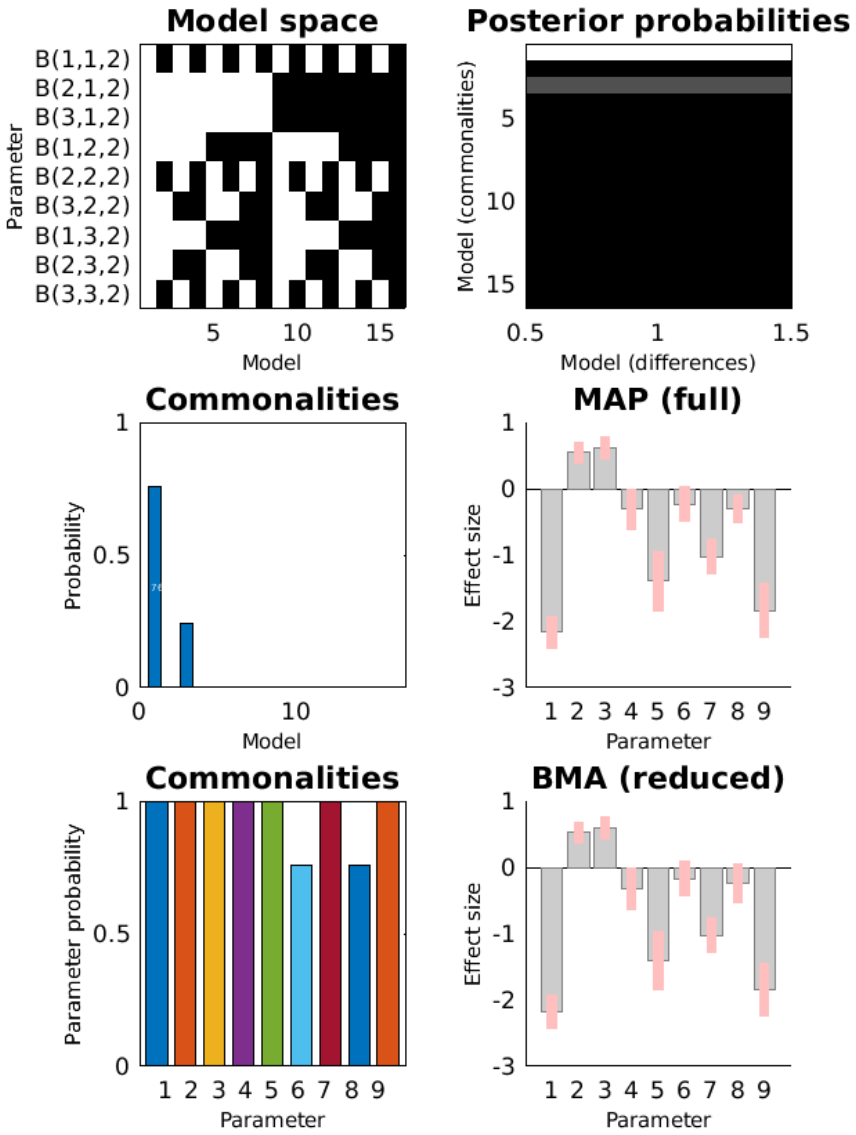

Figure 14: BMR for the 16 models in model space

We group these models into families by specifying an integer for each of the models in our model space defined in the GCM above. Models with the same integer belong to the same family. So to test whether between-region connections are needed or not, we assign all models with such connections to family 1, and the remaining two models (with modulation of self-connections only, or no modulated connections at all) to family 2. This can be done with this line of code:

```
families = [ones([1,14]), 2, 2];
```

We can then perform inference at the level of families by running:

```
[~, fam] = spm_dcm_peb_bmc_fam(BMA, BMR, families, 'NONE');
```

where, BMA and BMR are variables obtained from the BMR step earlier.[5] Running this produces Figure 15, with three panels.

The top left panel shows posterior probability of each family, where the family with modulation of at least one between-region connection has overwhelming probability (~1) compared to the family with no modulation of between-region connections. The top right panel shows posterior probabilities of each model conditioned by the probability of each family. Comparing this panel to the middle left panel from the previous figure shows how the probabilities have reduced after conditioning. The bottom left plot shows the grouping of models under each family.

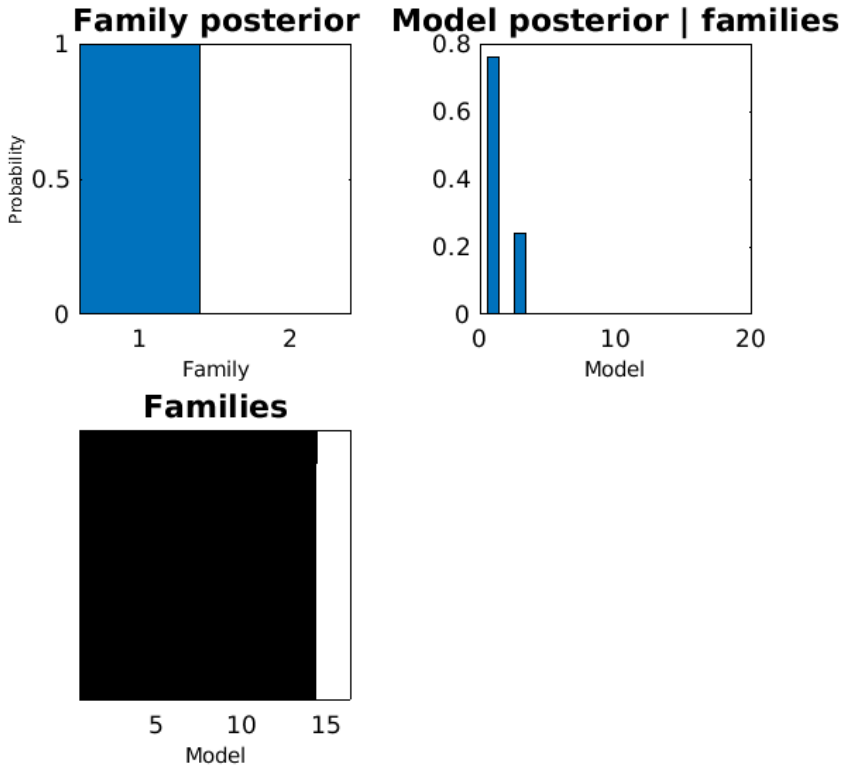

Figure 15: Family-wise comparison for modulation of between-region connections due to faces.

4.7.3.2. Family-BMC for forward connections

Since we have evidence for modulation of between-region connections, we can now test whether particular sub-groups of between-region connections are modulated by Faces. We do this for both forward, backward and lateral connections, by specifying families for each of them.

[5] Note we are assuming that each model is equally likely a priori. Note also that `spm_dcm_peb_bmc_fam` can return an updated BMA structure if needed, but we have ignored this output (using the "~" symbol in Matlab and passing in the argument 'NONE'), because we want to re-use the original BMA for further family comparisons below (or else you could specify the output as, e.g., "BMAf", so it does not overwrite the "BMA" from the full PEB that we will re-use below).

Repeat the last step in the previous section, except with the '`families`' variable now re-defined to `[ones([1,8]), 2*ones([1,8])]`. This divides the models into families with versus without modulation of forward connections. Then run again:

```
[~, fam] = spm_dcm_peb_bmc_fam(BMA, BMR, families, 'NONE');
```

This will produce results for the family-wise comparison showing overwhelming evidence (~1) for the family with modulation of at least one forward connection.

4.7.3.3. Family-BMC for backward connections

Similarly, to test backward connections, set the '`families`' variable to `repmat([1, 2, 1, 2], [1,4])` and re-run the `spm_dcm_peb_bmc_fam` command, and this should produce overwhelming evidence (probability ~1) for the family with modulation of backward connections.

4.7.3.4. Family-BMC for lateral connections

Finally, to test lateral connections, set the '`families`' variable to `repmat([1, 1, 2, 2], [1,4])` and re-run the `spm_dcm_peb_bmc_fam` command. This should produce moderate evidence (probability ~0.76), which may not be sufficient evidence (depending on one's a priori threshold), for the family with modulation of lateral connections.

4.7.3.5. Family-BMC for self-connections

Lastly, we can test whether self-connections are modulated by faces, by defining the '`families`' as `repmat([1, 2], [1,8])`. Re-running the `spm_dcm_peb_bmc_fam` call should produce the results showing overwhelming evidence for the family with self-connections.

4.7.4 PEB with subject-level covariates

In this section, we demonstrate the addition of an age covariate for 2nd-level PEB estimation. Although we do not expect any effect of age on modulation of connections in these data (given the narrow range of adult participants from 23 to 31), we conduct this exercise to highlight the key steps involved, since PEB was designed for testing differences between subjects (e.g, patients versus controls).

In a new batch, add a module to specify PEB by selecting '`SPM`' → '`DCM`' → '`Second level`' → '`Specify / Estimate PEB`'. In the options for this module, set '`Name`' to '`Age`' and select '`GCM_Full.mat`' estimated earlier from '`base_dir/fits/batch_gui/fmri`' for '`DCMs`'. Leave the '`DCM index`' option as is.

For '`Covariates`', select the option '`Specify covariates individually`' from the grey box below. (Alternatively, a full design matrix with all covariates can be passed via this option). Then click on '`New: Covariate`' in the grey box to create a pair of options – '`Name`' and '`Value`' for the covariate. Set '`Name`' to '`Age`'. For '`Value`', enter the following numbers, one on each line in the text box that pops open on clicking '`Specify`':

```
4.6, -1.4, 3.6, -0.4, -3.4, -0.4, 4.6, -0.4, 2.6, -3.4, -2.4, -2.4, -1.4, -2.4, 3.6, -1.4
```

These numbers are the ages of the 16 subjects taken from the BIDS 'participants.tsv' file (available here: https://openneuro.org/datasets/ds000117/versions/1.0.5/file-display/participants.tsv), after subtracting the mean age (and to one decimal place). Instead of entering the numbers manually, this 'participants.tsv' can be first loaded into MATLAB's workspace by running:

```
data = spm_load('participants.tsv');
```

Then in the batch interface's text box for entering values, simply enter this on the first line:

```
round(detrend(data.ages(1:16), 0), 1)
```

This will take ages of the 16 participants, remove their mean and round to 1st decimal place. Leave the remaining lines empty and press '`OK`'. The batch interface will evaluate the specified function and automatically fill in mean-corrected ages for all 16 participants.

This age covariate adds a second column to the design matrix (with the default first column still representing the group mean, and the mean correction of ages ensuring the two regressors are orthogonal, to ease interpretation).

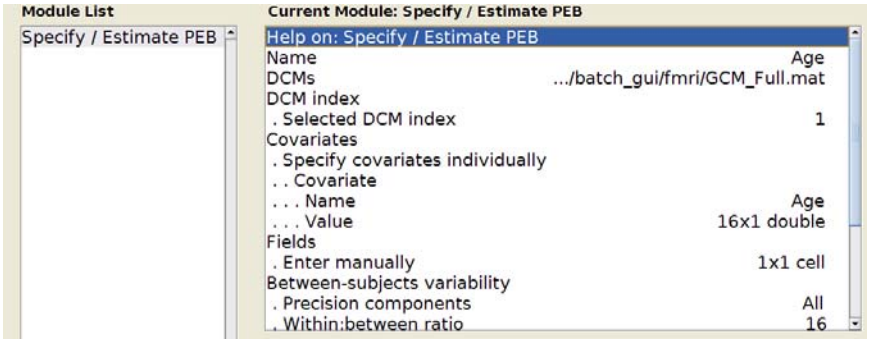

Figure 16: Specification of Covariates in PEB

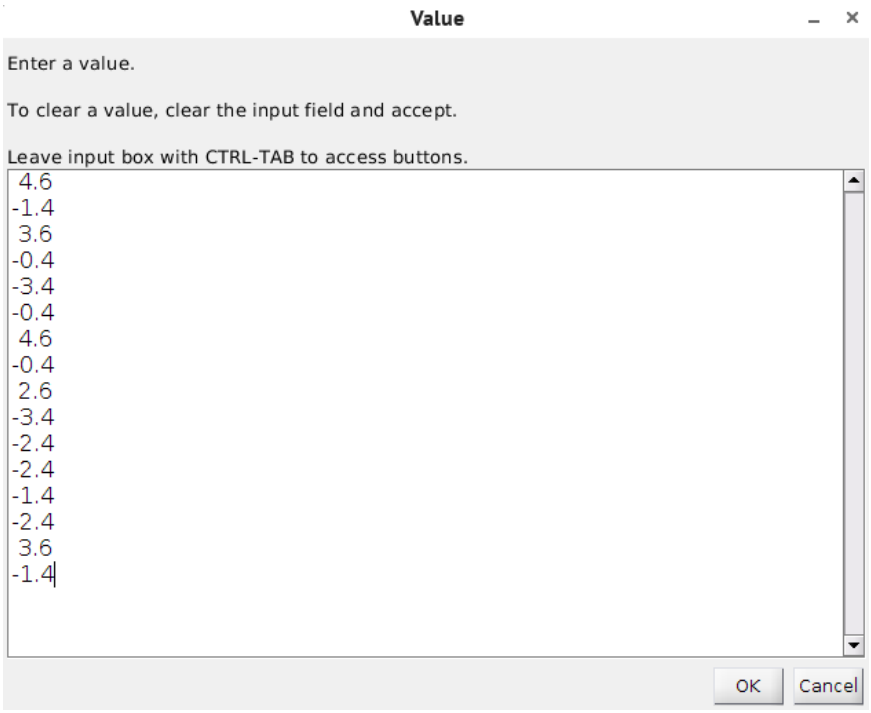

Figure 17: Entry of Age covariate values

In '`Fields`', select '`Enter manually`' and specify the field as '`{'B'}`', including curly braces. Lastly, set '`Review PEB parameters`' to '`Yes`', and press the green play button to estimate this PEB. This will generate the file '`PEB_Age.mat`' in the folder '`base_dir/fits/batch_gui/fmri`' and open the review window with estimates of group-level PEB parameters (Figure 18).

The review window now shows our design matrix in the top-left corner, with age as the second covariate. Since age was mean-centred, the first covariate represents mean modulation of connections across subjects. In addition to these common effects estimated at the group level ('`Second-level effect – Commonalities`'), the review window now has an additional option of viewing the effect of

age by selecting '`Second-level effect - Age`' from the drop-down menu. Thresholding these parameters based on posterior probabilities shows a subset of modulations were influenced by the age of participants. We do not have any hypotheses about these age effects, so do not discuss further, but if one wanted to perform further inferences about age, one could use the same model comparison methods described above, but on this second PEB covariate instead.

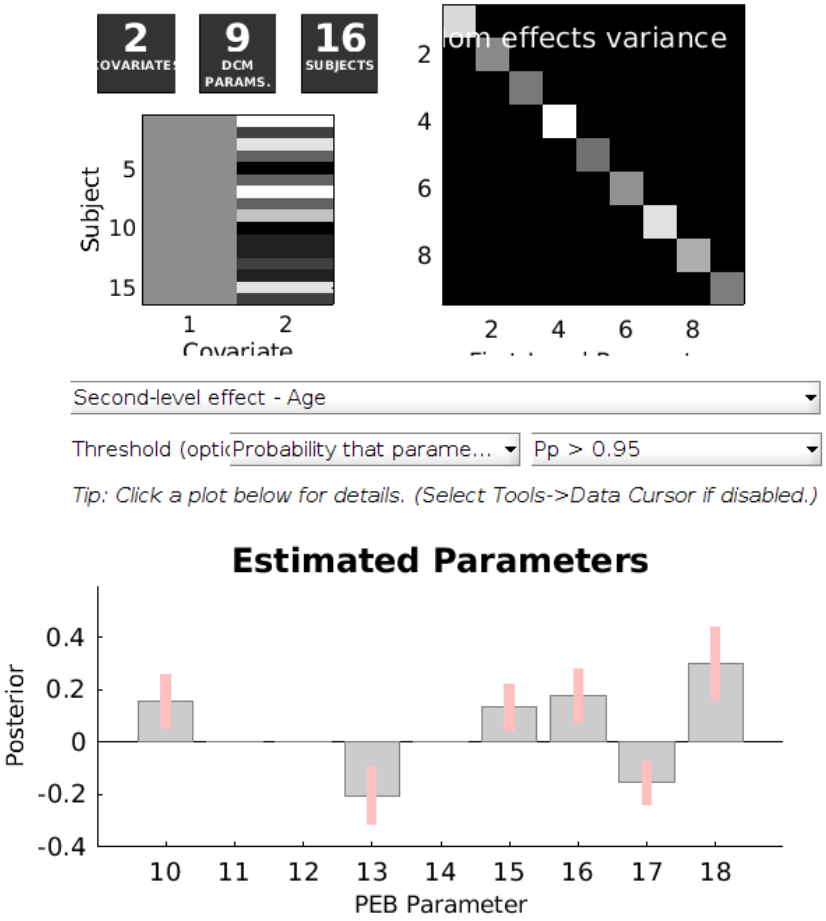

Figure 18: Review of PEB Parameters: 2nd level effect of Age

5. Discussion

We have demonstrated workflows for estimating PEB models for group-level inference on the connectivity parameters from DCMs of fMRI data. These workflows make use of SPM's graphical batch interface, and illustrate a systematic pipeline that begins with processed multimodal data for multiple subjects and ends with group-level inference about modulation of connections. We also show how to translate all steps of this pipeline from the graphical interface to batch scripts, which allow for greater flexibility and easier chaining of multiple dependent steps. This tutorial accompanies a similar one for M/EEG, for which notable differences exist in specifying individual DCMs, but the workflow for group-level PEB inference (i.e, once a GCM array has been specified) is identical.

Using our pipeline, we performed model comparisons at the group level on DCMs estimated from 16 subjects to test our hypotheses about effective connectivity between three ROIs (bilateral visual cortex and left and right fusiform face areas) during perception of faces versus scrambled faces. We first performed an automatic 'greedy' search over a model space consisting of nested versions of a 'Full' model, which featured modulation of all connections due to faces. The nested models were derived automatically by first identifying which parameters contribute to model evidence, and then switching all possible combinations of those parameters 'ON' or 'OFF'. In our case, 7 of 9 possible parameters were identified, and 128 nested models were derived from those parameters using this procedure. Posterior estimates and model evidence for these nested models were then analytically computed from the 'Full' model by a technique called Bayesian Model Reduction or BMR. Lastly, a model evidence-weighted combination of these 128 'reduced' models was used to estimate a Bayesian Model Average or BMA. We inspected this averaged model and identified dominant patterns of parameters that contribute to model evidence. Applying a threshold of 95% on the posterior probabilities of parameters in the BMA, we found that modulation of both forward connections from bVC to FFA, all three self-connections and the backward connection from rFFA to bVC were needed to explain the data.

While the automatic search is effective at pruning out parameters that do not contribute to model evidence, it operates at a very granular level by comparing individual parameters in the nested model space. This often leads to a large number of models being compared, which can lead to dilution of evidence. Moreover, such a finer granularity in the model space may not necessarily correspond to the hypothesis being tested. Alternatively, this model space can be constructed with a granularity that aligns closely with our hypotheses of interest, and then partitions of this model space can be systematically compared to identify which sets of parameters are needed in the model to explain the data. We used this procedure, called Family-wise Bayesian Model Comparison or FBMC, to test our hypotheses about specific types of connections (i.e. forward, backward, lateral and self). We created multiple nested models with different combinations of connections, grouped them under 'families', and then compared these families of models. Each definition of families corresponded to a hypothesis. For example, to test the hypothesis of whether modulation of forward connections (from visual cortex to left and right fusiform areas) is needed, we grouped all models with at least one forward connection in DCM's B matrix into one family, and all models without into another family. By performing FBMC across these families, repeating this process, we observed overwhelming evidence favouring the modulation of forward, backward and self-connections, but only moderate evidence supporting the modulation of lateral connections.

These findings from our DCM analysis suggest that an increased flow of information from bVC to FFA drives the increased response to faces over scrambled images. This preliminary evidence favouring a role of self-connections and forward connections from bVC to FFA are partly in agreement with extensive experimental findings in humans and non-human primates. The results differ somewhat from the companion tutorial on DCM for ERPs (using M/EEG data), where modulation of forward, backward and lateral connections were needed and no self-modulation was needed, but the underlying neuronal models and timescales are quite different, such that there are

good reasons why the connectivity captures different aspects of true neuronal interactions. Note we do not make strong scientific claims from these results, and other work has shown the importance of considering other ROIs, such as input to both EVC and FFA from early visual cortex (Lee *et al.*, 2022).

6. Appendix

6.1 Re-parametrising the SPM model

In the Supplementary Section 2 of our previous tutorial (Henson *et al.*, 2019), we fit a GLM to each subject in which there were 3 conditions: famous faces, unfamiliar faces and scrambled faces. In the present tutorial, we ignore the distinction between famous and unfamiliar faces, simply to make the tutorial simpler. More importantly, DCM normally assumes a common driving input (the '`C`' matrix in DCM), plus one or more modulatory inputs according to different conditions (the '`B`' matrix or matrices). To define the driving input, we need to define a new condition which contains all trials (i.e, onsets for all famous faces, unfamiliar faces and scrambled faces). Then the second condition will be faces only (i.e, onsets for famous and unfamiliar faces), and this will be used as the only modulatory input for DCM.

A second change to the GLM is to concatenate data across all 9 runs. In principle, DCM could be estimated for each run separately (and combined with 3-level PEB model across runs and subjects), but it is easier (and more typical) to concatenate runs within each subject. This requires concatenating volumes, onsets and motion parameters across runs, and creating a new `SPM.mat` file that contains only a single run, while maintaining the temporal filtering from the original run-specific design matrix (using '`spm_fmri_concatenate`' function below).

These steps can be achieved in two ways.

6.1.1 Updating existing SPM.mat files using MATLAB

The first approach is to run lines 171 to 235 from the section '`Combine`' in the script `spm_master_script_dcm_fmri_peb_batch.m`, which extracts the information from the `SPM.mat` files assumed present if you have run Supplementary Section 2 of Henson et al (2019). This code aggregates volumes, movement parameters and trial information across all runs for each subject, specifies a concatenated GLM with new conditions '`All`' and '`Faces`', and then estimates this model to an updated SPM.mat file.

6.1.2 Recreating SPM.mat files using SPM's Batch

The second is to use SPM's batch script. This creates new SPM.mat files from scratch (like in Henson et al., 2019), assuming you have the preprocessed fMRI volumes (either from Henson et al., 2019, or from Figshare link in main paper). This uses two batch jobs, `batch_stats_fmri_concatenated_specify_job.m` and `batch_stats_fmri_concatenated_estimate_job.m` in '`base_dir/code/fmri`'.[6]

[6] Alternatively, the batch job for specification can be created by editing "batch_stats_fmri_job.m" in https://figshare.com/collections/Multimodal_integration_of_M_EEG_and_f_MRI_data_in_SPM12/4367120 in the code/scripted folder, as follows:

1. Delete lines 17-64, ie from "matlabbatch{1}.spm.stats.fmri_spec.sess(2).scans = '<UNDEFINED>';" to "matlabbatch{1}.spm.stats.fmri_spec.sess(9).hpf = 128;", because we only want one session (run)
2. Delete lines 72 onwards, ie from "matlabbatch{2}.spm.stats.fmri_est.spmmat(1) = cfg_dep('fMRI model specification: SPM.mat File', substruct('.','val', '{}',{1}, '.','val', '{}',{1}, '.','val', '{}',{1}), substruct('.','spmmat'));"
3. Save as "batch_stats_fmri_concatenated_specify_job.m" in "code/fmri" directory

Unfortunately `spm_fmri_concatenate.m` is not batched, so see lines 237 to 291 of the supplied script '`spm_master_script_dcm_fmri_peb_batch.m`', which shows how to integrate this step as a part of the batch scripting pipeline.

The batch job for estimation can be created by starting SPM (eg '`spm fmri`' once SPM12 on Matlabpath), open Batch and create a new batch job to estimate this new concatenated SPM and evaluate a basic effects of interest contrast, as explained below:

From batch interface, select `SPM:Stats:Model Estimation`, and then `SPM:Stats:Contrast Manager`. For '`Select SPM.mat`' from Contrast Manager, select '`Dependency`' and `SPM.mat` file that (will be) produced by `Model estimation`.

In `Contrast Sessions`, select new F-contrast, called '`Effects of Interest`", and for '`Weights matrix`', enter '`eye(2)`', which is a MATLAB function for creating the identity matrix `[1 0; 0 1]`.

Next, add a T-contrast called '`All > Baseline`', and for '`Weights matrix`', enter `[1 0]`. (This is used in main paper to restrict voxels in the VOI to those activated by the stimuli.)

Then call batch in (par)for loop for each subject as per lines 242 to 291 of the section '`Combine`' in the script '`spm_master_script_dcm_fmri_peb_batch.m`'

While the onsets and regressors in the GLM have been concatenated, 'underneath the hood', SPM has kept separate highpass filters and separate autocorrelation estimation for each run (since not really a continuous timeseries, though small discontinuities across run boundaries might remain in the GLM, for example if the HRF for one run overlaps next, so it is a good idea to have ~30 seconds of rest at end of each run if planning to concatenate).

Finally, if you do not wish to use either step above, you can download re-parametrised SPM.mat files from: https://doi.org/10.6084/m9.figshare.25192793.v1

6.2 VOI creation

As stated in main paper, our ROIs are bilateral OFA and FFA. They are defined by 10-mm radius of spheres centred on coordinates explained in the main paper. You'll need to have re-parametrised the `SPM.mat` files as explained in Appendix 6.1.

From batch interface, press `SPM:Util:Volume of Interest`, then:

- Enter '`1`' for '`Adjust data`' (for selecting the "effects of interest" contrast above, the null space of which defines the confounds that will be removed from the data, e.g. motion parameters)
- Enter '`1`' for '`Which session`' (since there is only one run now runs have been concatenated)
- For '`Region(s) of Interest`', select '`New: Sphere`', enter '`10`' for '`Radius`', leave '`Centre`' '`undefined`' (we will complete below).
- In '`Region(s) of Interest`' again, select '`New: Thresholded SPM`', and set Contrast to '`2`' (i.e, we only want voxels that are activated by stimuli versus baseline)
- For Expression, enter '`i1&i2`'[7]
- So empty fields are `SPM.mat` file, `Name of VOI`, and `Centre` (which we provide in script).
- Save and script as '`batch_VOI`'

[7] This is the same format as SPM's ImCalc function to do basic operations on every voxel across a set of images indicated by i1, i2, i3, etc. Here the use of "&" restricts voxels to those that are non-zero in both images i1 and i2 (where those images are the sphere and the set of voxels showing p<.001 uncorrected).

- Type in these coordinates as centres of spheres for each ROI:

  bVC [0, –90, 0]

  lFFA [–42, –56, –20]

  rFFA [+42, –52, –14]

Then call batch in (par)for loop for each subject as in lines 305 to 350 from the section '`VOI`' of the script '`spm_master_script_dcm_fmri_peb_batch.m`'. This will produce several graphical outputs that you can examine, but focus here is on DCM, and more details on VOI extraction can be found in SPM12 manual.

This saves several files in each subject's directory for each ROI (VOI), the most important of which are called '`VOI_bVC_1.mat`', '`VOI_lFFA_1.mat`' and '`VOI_rFFA_1.mat`'. These contain the fMRI timeseries (first singular temporal vector) for each ROI that DCM will fit below (plus some other parameters needed for DCM). The other files are images, e.g, '`VOI_lOFA_mask.nii`', which contains a binary image defining voxels within an ROI, and '`VOI_lOFA_1_eigen.nii`', where voxels contain the spatial weights instead (first singular spatial vector).

7. References

David, O. *et al.* (2006) 'Dynamic causal modeling of evoked responses in EEG and MEG', *NeuroImage*, 30(4), pp. 1255–1272. Available at: https://doi.org/10.1016/j.neuroimage.2005.10.045.

Henson, R.N. *et al.* (2019) 'Multimodal Integration of M/EEG and f/MRI Data in SPM12', *Frontiers in Neuroscience*, 13. Available at: https://www.frontiersin.org/article/10.3389/fnins.2019.00300 (Accessed: 25 June 2022).

Henson, R.N. and Mouchlianitis, E. (2007) 'Effect of spatial attention on stimulus-specific haemodynamic repetition effects', *NeuroImage*, 35(3), pp. 1317–1329. Available at: https://doi.org/10.1016/j.neuroimage.2007.01.019.

Lee, S.-M. *et al.* (2022) 'Effects of face repetition on ventral visual stream connectivity using dynamic causal modelling of fMRI data', *NeuroImage*, 264, p. 119708. Available at: https://doi.org/10.1016/j.neuroimage.2022.119708.

Litvak, V. *et al.* (2011) 'EEG and MEG Data Analysis in SPM8', *Computational Intelligence and Neuroscience*, 2011, pp. 1–32. Available at: https://doi.org/10.1155/2011/852961.

Moran, R., Pinotsis, D.A. and Friston, K. (2013) 'Neural masses and fields in dynamic causal modeling', *Frontiers in Computational Neuroscience*, 7. Available at: https://doi.org/10.3389/fncom.2013.00057.

Pereira, I. *et al.* (2021) 'Conductance-based dynamic causal modeling: A mathematical review of its application to cross-power spectral densities', *NeuroImage*, 245, p. 118662. Available at: https://doi.org/10.1016/j.neuroimage.2021.118662.

The MathWorks Inc. (2018) 'MATLAB R2018a'. Natick, Massachusetts: The MathWorks Inc.

Wakeman, D.G. and Henson, R.N. (2015) 'A multi-subject, multi-modal human neuroimaging dataset', *Scientific Data*, 2(1), p. 150001. Available at: https://doi.org/10.1038/sdata.2015.1.

Zeidman, P., Jafarian, A., Corbin, N., *et al.* (2019) 'A guide to group effective connectivity analysis, part 1: First level analysis with DCM for fMRI', *NeuroImage*, 200, pp. 174–190. Available at: https://doi.org/10.1016/j.neuroimage.2019.06.031.

Zeidman, P., Jafarian, A., Seghier, M.L., *et al.* (2019) 'A guide to group effective connectivity analysis, part 2: Second level analysis with PEB', *NeuroImage*, 200, pp. 12–25. Available at: https://doi.org/10.1016/j.neuroimage.2019.06.032.