fMRI Basics: Single Subject Analysis

This session is intended to give an overview of the basic process of setting up a general linear model for a single subject. This stage of the analysis is also variously referred to as the first level model, fixed effects model, or single subject fixed effects model. A typical study would involve creating a first level model for each subject, and then entering all the results into a group analysis (also called random effects or second level model).

The data we'll use in this session comes from a localiser task carried out during an experiment designed to investigate differences between consciously perceived and undetected faces. Participants completed 3 sessions of the localiser task alternating with 3 sessions of the main task. Within each session of the localiser task they saw 2 blocks of face stimuli, 2 of scrambled face stimuli, and 2 blocks containing pictures of everyday objects. Each block lasted for around 16 seconds and contained 20 pictures, each of which was presented for 800ms. The blocks were presented in a random order and alternated with 16 second blocks of fixation.

The workshop is based around a batch script, so the first thing is to create a local copy and open it with the Matlab editor:

```
cp       /imaging/russell/MRI_workshop/single_subject_fixed_effects.m
/imaging/(username)/MRI_workshop
```

You'll also need a copy of the file that contains information about the timings of experimental conditions:

```
cp      /imaging/russell/MRI_workshop/Events/block_design_timings.txt
/imaging/(username)/MRI_workshop/CBU071112
```

This is simply a tab delimited text file containing 4 columns of data. The first column contains the session number, the second the condition number, the third the onset times of each condition (in ms relative to the start of a session), and the fourth the duration of the condition. This text file will be read into Matlab by the analysis script, and contains all the information about the experimental design necessary to create a design matrix.

The details of event types and timings are usually generated by your stimulus presentation program, and either saved directly in the format used by your analysis script, or formatted at a later stage (e.g. if you use e-prime, which saves output in a standard format). Stimulus presentation is synchronised with your fMRI data by collecting pulses sent out by the scanner at the beginning of each volume and received by the stimulus presentation computer. Most people at the CBU use Rhodri's scannersync routines (http://imaging.mrc-cbu.cam.ac.uk/methods/ScannerSync) to do this. The easiest way to synchronise your experimental presentation is to start your programme running, then have it wait until the first pulse is detected. The scannersync "StartExperiment" command will do this, and will also start its internal timer at this point. Throughout the experiment you can then get the times of any events / responses etc relative to the onset of the first scan using the GetTimer command. This is the minimum amount of information necessary to synchronise your experiment, but scannersync can also handle a number of other operations, including waiting for particular pulse numbers etc. SPM will accept event onset times in either seconds or scans.

It is also usual to collect several dummy scans at the start of each session prior to the first stimulus to allow for T1 equilibriation effects. These are discarded from any further analysis, and event timings adjusted accordingly.

Running the script

Once you've opened the script, change usr_name on line 12 to your own user name then save the script. Put a break point on line 12 and press F5 to begin running the script. As in the previous session, we'll step through the script one line at a time by pressing F10. The script is divided into several sections:

Setting up and defining variables:

Lines 7 to 69

Here we specify various bits of information necessary to run the analysis. The first part (lines 20-30) defines the location of various files and directories, including the location of the image files, a directory to use for the results of the analysis, the location of the timings file, and the location of the file saved by the realignment procedure that contains the movement parameters.

The next part (lines 37-67) define several other parameters – see the comments in the script for further details.

Defining contrasts:

Lines 77 to 115

This section defines 2 contrasts that will be evaluated after the model has been estimated. There are clearly a number of other contrasts that might also be of interest – feel free to add any more contrasts you want to include here. Contrasts can also be defined later using the results ui.

The information that needs to be defined for each contrast is a name, a type (T or F), and a vector defining how to weight each column in the design matrix.

Even for a design matrix with a smallish number of columns like the one we'll create here, it can be tedious (not to mention being error prone) to type in long strings of zeros and ones. One advantage of using a script is that it allows you to build up contrasts bit by bit, and to re-use various chunks of information. For example, here the script defines a vector of 6 zeros that can be used to cover the columns that represent the movement parameters. This vector can then be used when defining the contrasts, which saves having to type [0 0 0 0 0 0] 3 times for every contrast. The contrasts here are defined in full, but since the order of events in each session is identical, the contrast vectors could also be defined using functions such as repmat, e.g.

```
contrast_vector = [repmat([1 -1 0 mz],1,n_sessions) sz]
```

Building the SPM data structure:

Lines 123 to 237

SPM tends to work by storing the values of the parameters necessary for particular functions as fields in large Matlab data structures. In the pre_processing script, you saw that lots of the functions needed “options” structures containing several different values. In general, a lot of SPM batch scripting ends up being about creating these data structures, filling in the correct values, and passing them to the correct functions.

One of the most detailed data structures is the one used to define the models for statistical analyses. In deference to convention, this structure is called “SPM” in the current script, and we’re now at the point to start filling in various fields. Within the script, the structure could be given any name, but the name of the fields within the structure are fixed as the spm functions are expecting them to have specific names.

As we go through the script, we’ll gradually fill in the necessary fields. We then pass the structure to various SPM routines which use these values to set up the model. At the end of the whole process, the data structure gets saved to your analysis directory as SPM.mat. This can be reloaded into Matlab at any time, and contains all the details of your analysis. More information about the different fields of the SPM structure can be found by typing `help spm_fmri_spm_ui` and `help spm_spm`.

Basis Function - Lines 131 to 157

First we create a structure called xBF which holds information about the basis function we’re going to use. Most of the parameters are described in the comments in the script. It is also worth noting that the units specified in xBF.Units define the units that will be used for timings throughout the model, so your event times should be in the same units specified here.

Miscellaneous parameters - Lines 161 to 166

Line 164 enters the TR into the correct field. Line 165 specifies whether each image volume should be scaled to have the same overall mean value or not – this is largely a legacy of PET analyses where the overall signal would decline throughout the experiment. Generally its set to “none” for fMRI as mean scaling each volume can create its own problems (e.g. if some volumes contain a large activation, this will increase the overall mean signal, and mean scaling the whole volume will tend to produce a relative decrease the signal in areas outside the activation cluster). Line 166 specifies whether spm should try to estimate the auto-correlation present in the residual variance.

Event timings and images - Lines 170 to 230

Next stage is to collect information about the experimental design. This is done session by session using a loop. First, the images that make up the data for each session are collected (line 180) and appended to a master list of all the images (line 181). Images are collected one session at a time as this provides an easy way of counting the number of images in each session, another parameter that spm needs (line 183). We also keep a cumulative total of the images and store the number of the first and last images making up each session (189-90). These are needed later on to select the correct rows from the matrix of movement parameters (e.g. see line 224), as this is not split into separate sessions.

After this, the script loops through each of the experimental conditions (195-216). For each condition the correct rows are selected from the timings matrix by finding rows where the first column matches the current session number and the second column matches the current condition. This gives an index indicating which rows to use (condition_index, line 196). Onset times are obtained by selecting the correct rows from column 3. These are converted from ms to s, and adjusted to take account of the number of dummy scans (197). Similarly, if event durations are to be explicitly modelled, the correct durations are retrieved from column 4 of the timings matrix, otherwise all events are given a duration of zero (203-7). Finally, information about each condition is stored in the SPM structure (211-5).

The information in SPM.Sess.U is used to create regressor functions for the experimental conditions. Firstly a timing function is created where the onset of each condition or event is represented as a finite impulse. The timing function is also sometimes called a stick function as when it's plotted on a graph with time on the x-axis, it looks like a series of thin vertical lines (sticks). It's at this point that spm makes a distinction between events (conditions where a zero duration is specified) and epochs (conditions where duration is explicitly modelled). Events are modelled as a single stick with height equal to xBF.T, whereas epochs are modelled as a series of sticks each of height 1 and spanning the entire duration of the epoch. Epochs can be interpreted as giving an estimate of the response per unit time, whereas events give an estimate of the response as if the condition occurred instantaneously. This difference between events and epochs means that the 2 are not directly comparable. You can model some conditions as events and some as epochs, but you should avoid contrasts directly comparing the 2. Finally, the timing functions are convolved with the basis function to create the regressor functions.

After entering the high pass filter cutoff (218), any additional covariates can be defined (223-9). In this design the main covariates we might want to include are the movement parameters, but in practice any covariates can be included here. The values for the covariates are entered by including a matrix of values that has one row for each scan in the current session and one column for each covariate (e.g. line 224). In contrast to the event timings in SPM.Sess.U, the covariate values in SPM.Sess.C are simply entered directly into the model. Both are ultimately estimated as part of the same model however, and you could if you wanted to manually create the regression functions for the experimental conditions and enter them as covariates (e.g. if you wanted to model different conditions with different basis functions, something that isn't handled by the standard spm code).

After all sessions are completed, the master list of image files is entered into the SPM structure (234).

Creating the design matrix:

Lines 242-247

Once the necessary information has been defined, the SPM structure can be passed to spm_fmri_spm_ui. This uses the information to create the design matrix and also performs a couple of other functions such as reading the headers of each of the image volumes specified in SPM.xY.P (using a function called spm_vol) and calculating the global mean signal in each volume. This usually takes a minute or so to run (depending on the size of the design and the load on the Linux cluster), and when it has finished, it displays a graphical representation of the design matrix in the graphics window. The function also returns a modified version of the SPM structure that contains extra information about the design.

Estimating the model:

Lines 253-258

At this point the SPM structure contains all the information necessary to estimate the design. This is the core of the whole process, and is set in motion by what has to be one of the best lines of code anywhere:

```
SPM = spm_spm(SPM)
```

As part of the estimation, spm_spm creates a series of "beta" images. One image is created for each column in the design matrix, and the value at each voxel within these images represents the beta value for that predictor at that voxel.

Also created are an image that contains the error term (i.e. the residual or unexplained variance) for each voxel (ResMS.img / hdr), a mask image defining which voxels are included in the analysis (mask.img / hdr), and an image defining the effective spatial resolution of the images (RPV.img / hdr). In the latter, RPV stands for "resels per voxel" and is used when evaluating statistical thresholds that correct for multiple comparisons (more on this below).

By default, SPM determines which voxels are inside the brain, and should therefore be included in the analysis, by calculating the global mean signal for each volume and labelling all voxels with a signal of less than 1/8th of this signal as outside the brain. Voxels that fall below this threshold in *any* of the images are excluded, as are voxels that have a constant value across all images.

Unless there is a particularly heavy demand on the system, the current model should take less than 5 minutes to estimate.

Creating the contrasts:

Lines 264 to 280

Having estimated the model, the final step is to generate the contrast images. Firstly (lines 270-6) the information we defined above is passed to spm_FcUtil. This checks whether the contrast is legitimate, and if so returns a data structure containing additional information about the contrast. This is stored in SPM.xCon. The complete SPM structure is then passed to spm_contrasts which calculates the values for the contrast. For t-contrasts, this process creates a series of contrast images ("con*.img" / "con*.hdr" pairs) and a corresponding series of statistic images (spmT*.img). The equivalent files for F contrasts are ess*.img and spmF*.img.

As described below, the contrast results can be viewed using the results ui.

Reviewing a design matrix

At any point after the design matrix has been created (i.e. after spm_fmri_spm_ui has been run), the design matrix can be reviewed by clicking “Review” in the menu window. This opens a file selection box – select the SPM.mat file containing the design you want to reveiw, click “Design” in the menu bar that appears at the top of the interactive window, then select “Design Matrix”. A graphical representation of the design matrix appears in the graphics window.

The review function can also be used to see the design orthogonality matrix (i.e the matrix showing the correlations between the columns of the design matrix), and also to explore the regressors used for each of the experimental conditions.

The design orthogonality matrix is useful, but be careful as it only shows first order correlations between individual pairs of variables. Correlations between linear combinations of predictor variables (e.g between [columns 1+2] and [columns 3+4]) aren’t shown (but can be equally problematic for the design).

Going Fishing

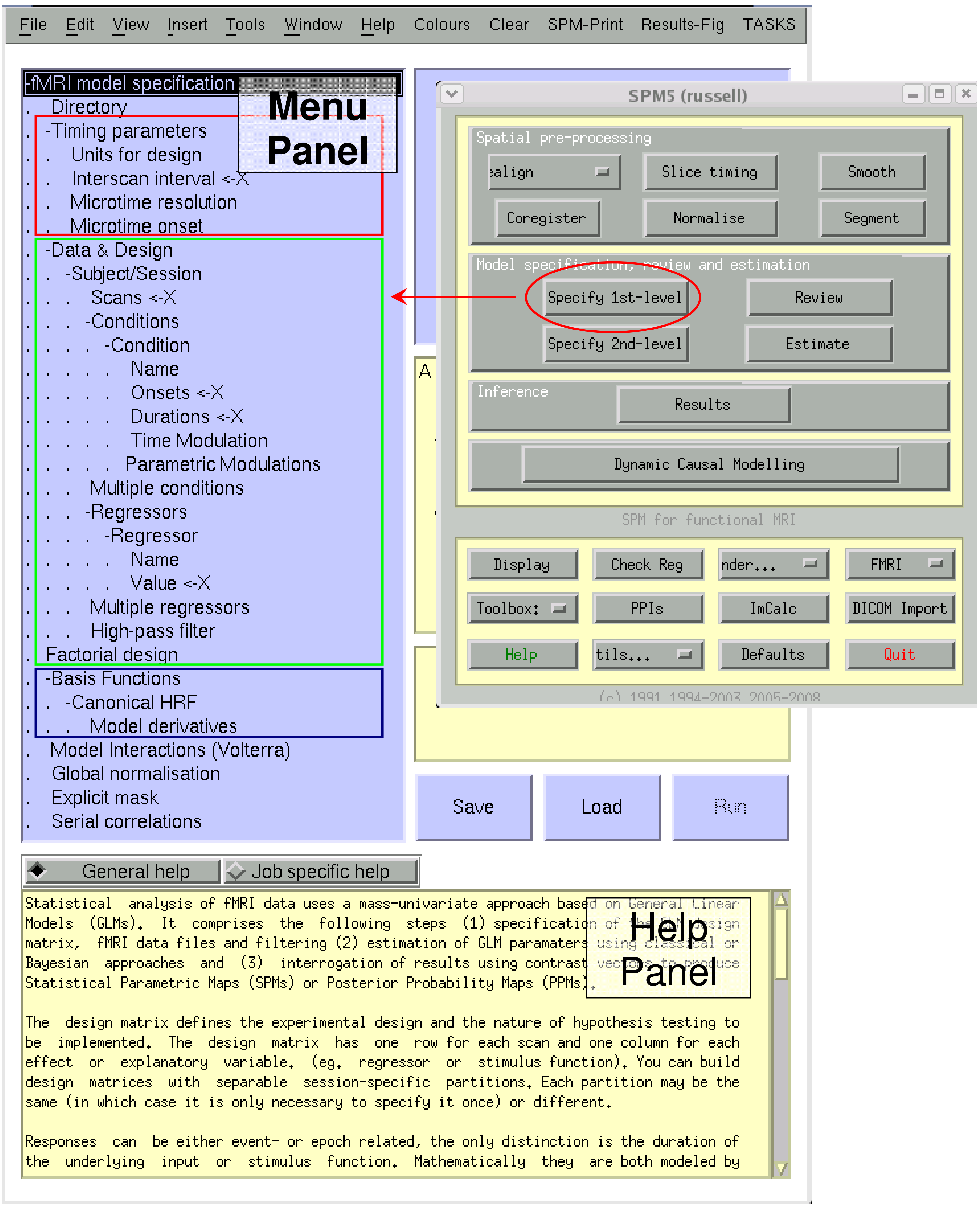
As you’ll appreciate by now, there are a lot of arbitrary (or at least semi-arbitrary) parameters involved in analysing an fMRI data set – which pre-processing stages you choose to implement, the amount of smoothing, the high pass filter value used, or the basis function set to name but a few. It is also often possible to construct the first level model in several different ways – for example including or excluding certain experimental conditions or covariates, combining or splitting experimental conditions into different predictor variables, modelling only the main effects in a factorial design vs. modelling every single cell, explicitly modelling event duration and so on.

The current design is relatively simple, but if you want to explore it further, some options include re-running it without the movement parameters – does that make any difference? What about changing the width of the smoothing kernel? Or using no smoothing? What about changing the high pass filter cutoff? You could also try using different basis functions.

There are also numerous other contrasts that could be created. For example, are there any areas where objects give greater responses than scrambled pictures? What about where faces give greater responses than objects (or vice versa)? Or areas that show variance in signal correlated with particular movement parameters? What about areas that are correlated with any movement?

First level model via the GUI

It’s probably worth having a look through the gui once or twice as it gives a good overview of all the parameters that can be specified in a first level model, and also often provides useful information about each of these parameters in the help panel.



Statistical analysis of fMRI data uses a mass-univariate approach based on General Linear Models (GLMs). It comprises the following steps (1) specification of the GLM design matrix, fMRI data files and filtering (2) estimation of GLM paramaters using classical or Bayesian approaches and (3) interrogation of results using contrast vectors to produce Statistical Parametric Maps (SPMs) or Posterior Probability Maps (PPMs).

The design matrix defines the experimental design and the nature of hypothesis testing to be implemented. The design matrix has one row for each scan and one column for each effect or explanatory variable. (eg. regressor or stimulus function). You can build design matrices with separable session-specific partitions. Each partition may be the same (in which case it is only necessary to specify it once) or different.

Responses can be either event- or epoch related, the only distinction is the duration of the underlying input or stimulus function. Mathematically they are both modeled by

Figure 1 - First level model gui with expanded menus

In the SPM menu window, click “Specify 1st level”. A new gui opens in the graphics window (see figure 1). This follows the same format we saw in the gui interfaces for the preprocessing routines: The top left panel contains a menu-like display showing which parameters need to be set, and the top right panel provides an interactive way to set those parameters, either by launching the file selector, showing a clickable menu of choices, or opening a text box where values (or evaluated expressions) can be typed directly.

The largest section is labelled “Data and Design” (outlined in green in the figure). This is where you specify all the event types / timings etc. Click “Data and Design” in the menu panel, then “New subject session” in the interactive panel (top right). A new “Subject/Session” subheading appears under Data and Design, double click it to open up the session menu. Click on each of the menu items to define the conditions and timings. These can be specified by loading a mat file that contains details of all the conditions (click “Multiple Conditions” then select the mat file), or one at a time, manually entering all the onset times etc (click “Conditions”, then “New Condition” in the interactive panel, then double click the new “Condition” subheading that appears underneath “Conditions”). The same goes for other covariates (called “Regressors” here), which can be loaded from a mat file (“Multiple Regressors”), or entered one at a time (“Regressors”, “New Regressor” etc).

Below “Regressors” is the “Factorial Design” section, which allows you to generate all the F contrasts for a factorial design. This works simply by specifying a number of factors, and the numbers of levels of each.

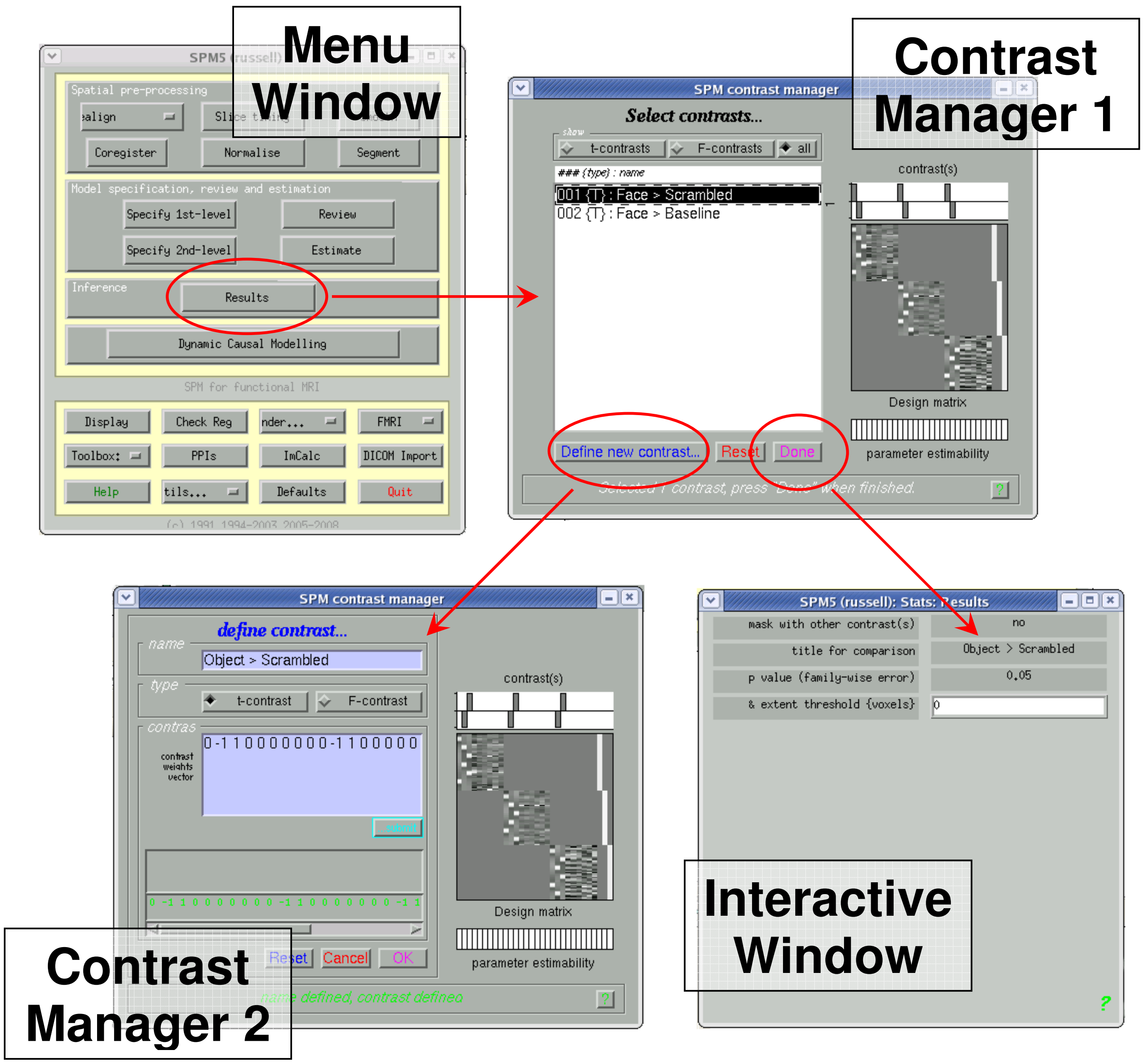
Next is the section where various different basis functions can be specified. Almost always this will be the canonical href (with the additional option of adding temporal and spatial derivatives), but the finite impulse response set can also come in useful if you want to investigate the time course of event related responses, or want to allow for the possibility of non-canonical haemodynamic responses.

The final 4 options allow you to specify non-linear interactions in the haemodynamic response to successive events (“Model Interactions (Volterra)”), to mean scale each volume in the time series (“Global Normalisation”), to specify which voxels to include in the analyis (“Explicit Mask”), and to model non-sphericity in the residual variance (“Serial Correlations”).

The Results Interface

Once you’ve estimated a design and created the contrasts, you’ll probably want to have a look at the results.

- To view the results of the contrast, click “Results” in the menu window (fig 2)
- This opens the SPM contrast manager window. On the left of the window is a list of the currently defined contrasts, and on the right is a picture of the design matrix. Click on the name of a contrast to select it. This also brings up a schematic representation of the contrast above the design matrix.
- To define a new contrast click “Define new contrast”
- This brings up a second contrast manager window. On the right are several boxes that let you define the name, type, and a vector of weights for the new contrast. When you’ve entered all the information, click “submit” and SPM will tell you



name: Object > Scrambled

contrast weights vector: 0 -1 1 0 0 0 0 0 0 0 -1 1 0 0 0 0 0 0

mask with other contrast(s)	no
title for comparison	Object > Scrambled
p value (family-wise error)	0.05
& extent threshold {voxels}	0

Figure 2 – Results interface and contrast manager

whether the contrast is legitimate. If it is, click OK to return to the first contrast manager window.

- Once you've selected or defined a contrast, click "Done".
- This closes the contrast manager, and opens a series of text boxes in the interactive window. These allow you to specify:
    - whether or not to mask the results with those of another contrast
    - a name for the contrast
    - *a statistical threshold* – only voxels which reach the specified p value will be displayed. The choice of threshold includes no correction for multiple comparisons (Unc), corrected for multiple comparisons using the false

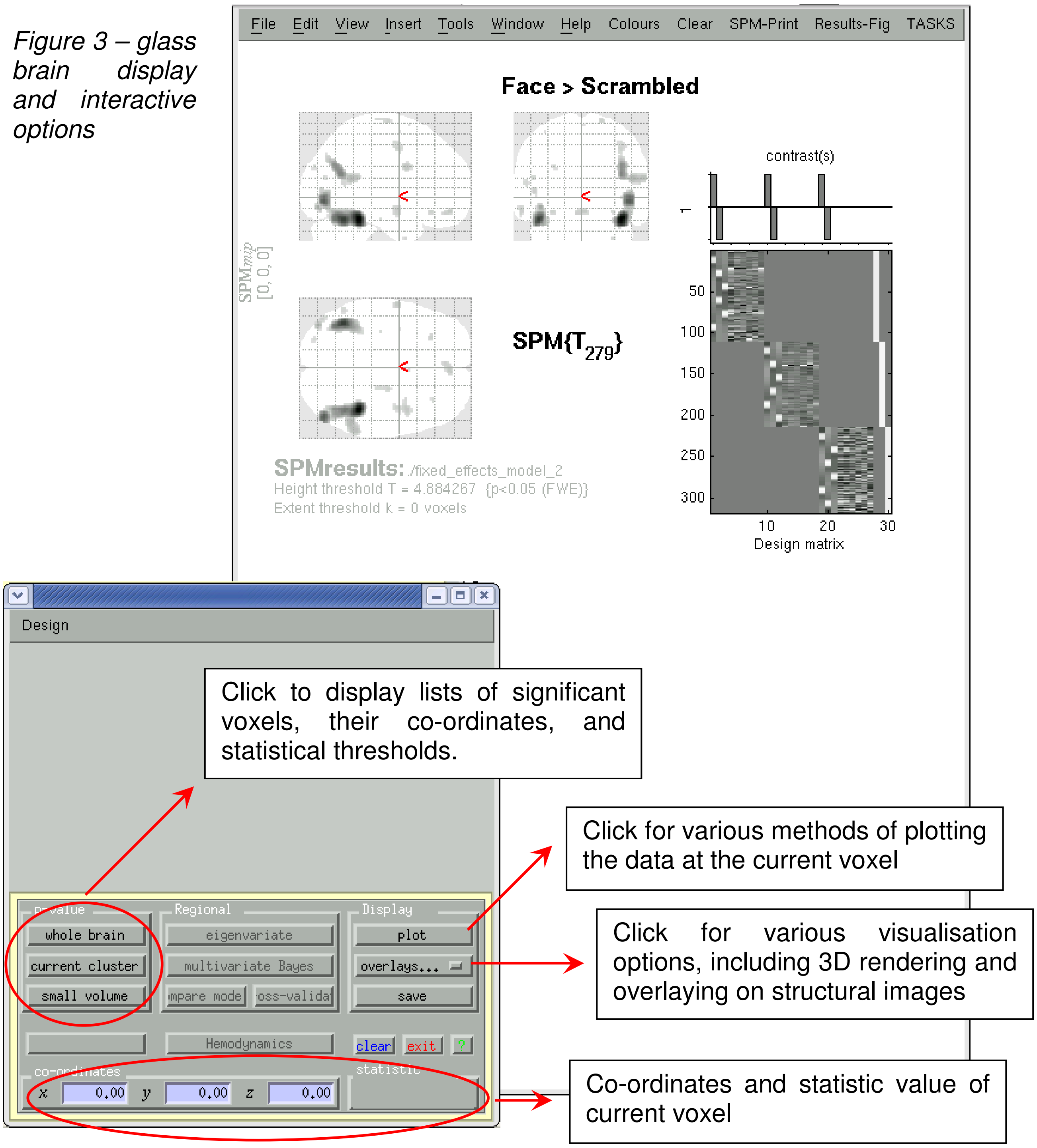
discovery rate (FDR), and corrected for multiple comparisons using the family wise error rate (FWE). Uncorrected thresholds ignore the fact that you're carrying out several thousand comparisons. FDR works by specifying the percentage of false alarms you are prepared to accept – e.g. if you find 1000 signifiant voxels using an FDR of 0.05 (5%), on average 50 of those voxels will be false positives. FDR is now widely used (mainly because it tends to be less conservative than FWE), but it does have some odd properties. FWE correction is conceptually the same as a Bonferroni correction, but uses the theory of random fields to determine the number of independent tests. The basic idea is that because the result at any voxel is not truly independent of the result in neighbouring voxels (especially after smoothing), the number of independent tests is lower than the number of voxels. Random fields theory is used to determine the number of "resels" (resolution elements) in the data, and the correction is made on the basis of this.
- *an extent threshold* – only clusters that contain more than the specified number of voxels will be displayed.

After you've entered the extent threshold, the result of the contrast appears in the graphics window (fig 3). This is shown in 3 orthogonal slices, the so called "glass brain" display (or maximum intensity projection). At the same time, a new gui appears in the interactive window (fig 3). This contains various options for visualising the results, for example 3D rendering, overlaying the results on an anatomical image, and various plotting options. The glass brain display is interactive, and you can click around the results to move the cursor to a particular voxel. The co-ordinates and statistic value at the current voxel are shown at the bottom of the interactive window. You can also right click the glass brain display for a context menu that lets you jump to the voxel with the highest overall value (global maximum), or the nearest local maximum).

To create a 3D rendering:
- Click "Overlays" in the interactive window, then "render" from the drop down menu
- A file selection box opens, navigate to /imaging/local/spm/spm5/rend and choose one of the files listed.
- A series of prompts now appear in the interactive window:
  - Style – old or new. Old uses the "hot iron" type of display, where voxels are rendered to the surface using a maximum intensity projection represents the statistical values (red for low through orange and yellow to white for high). New uses a single colour, but represents the depth of voxels below the cortical surface by manipulating the transparency of the colour (more transparent / less saturated = deeper)
  - If you select "new" you'll also be asked to specify how much to brighten the blobs by (how much to project subcortical clusters to the surface – "slightly" usually works well), and the colour in which to render the blobs.
- Rendering is performed by the function spm_render. Type help spm_render in the Matlab command window for more information.

To overlay on orthogonal sections :
- Click "Overlays" in the interactive window, then "sections" from the drop down menu.

Figure 3 – glass brain display and interactive options



Click to display lists of significant voxels, their co-ordinates, and statistical thresholds.

Click for various methods of plotting the data at the current voxel

Click for various visualisation options, including 3D rendering and overlaying on structural images

Co-ordinates and statistic value of current voxel

- A file selection box opens asking for a structural image to render onto. Select any structural in the same space as your statistical results. Various canonical images can be found in /imaging/local/spm/spm5/canonical (single_subj_T1.nii is a good one to use).
- Three orthogonal sections appear below the glass brain. These are navigable in the same way as the glass brain, and also linked to the glass brain display - moving to a new voxel on one display automatically updates the other.

- The display is performed by the function spm_sections.

To display a list of significant voxels :
- In the interactive menu, click “whole brain”.
- This brings up a list of co-ordinates underneath the glass brain.
- The results are divided into a number of clusters, and by default 3 voxels from each cluster are displayed.
- The list gives various statistics, including set level, cluster level, and voxel level p values.
- The list is interactive, and clicking on any of the co-ordinates automatically moves the cursor in the glass brain.
- The list is controlled by the function spm_list. This also outputs the table as a data structure which can then be formatted and saved to a text file (e.g. for inclusion in manuscripts).

In addition to the gui options, displaying a contrast also saves some variables to the Matlab base workspace. The most useful of these is a structure called xSPM. This contains details of all the significant voxels, their co-ordinates and statistic values etc. Once you’ve displayed a contrast, type xSPM in the Matlab command window to view its contents.