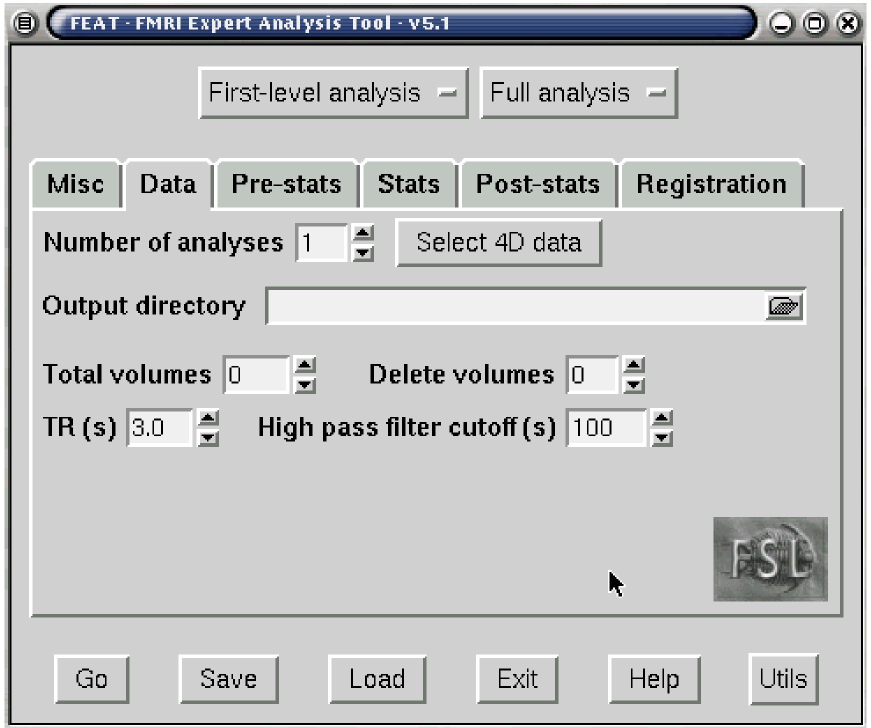
FEAT - FMRI Expert Analysis Tool

User Guide - FEAT Version 5.10
Copyright 2000-2003 Steve Smith, FMRIB, University of Oxford
[PDF version by Morten L. Kringelbach]



First-level analysis
Full analysis
Number of analyses 1
Output directory
Total volumes 0
Delete volumes 0
TR (s) 3.0
High pass filter cutoff (s) 100

Introduction

For other information on FEAT and updated journal references, see the FEAT web page. If you use FEAT in your research, please quote the journal references listed there.

FEAT is a software tool for high quality model-based FMRI data analysis, with an easy-to-use graphical user interface (GUI). FEAT is part of FSL (FMRIB's Software Library). FEAT automates as many of the analysis decisions as possible, and allows easy (though still robust, efficient and valid) analysis of simple experiments whilst giving enough flexibility to also allow sophisticated analysis of the most complex experiments.

Analysis for a simple experiment can be set up in less than 1 minute, whilst a highly complex experiment need take no longer than 5 minutes to set up. The FEAT programs then typically take 10-30 minutes to run (per first-level session), producing a web page analysis report, including colour activation images and time-course plots of data vs model.

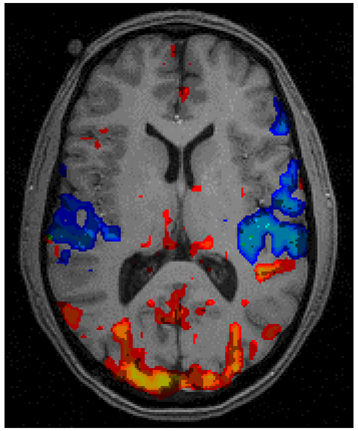
The data modelling which FEAT uses is based on general linear modelling (GLM), otherwise known as multiple regression. It allows you to describe the experimental design; then a model is created that should fit the data, telling you where the brain has activated in response to the stimuli. In FEAT, the GLM method used on first-level (time-series) data is known as FILM (FMRIB's Improved Linear Model). FILM uses a robust and accurate nonparametric estimation of time series autocorrelation to prewhiten each voxel's time series; this gives improved estimation efficiency compared with methods that do not pre-whiten.

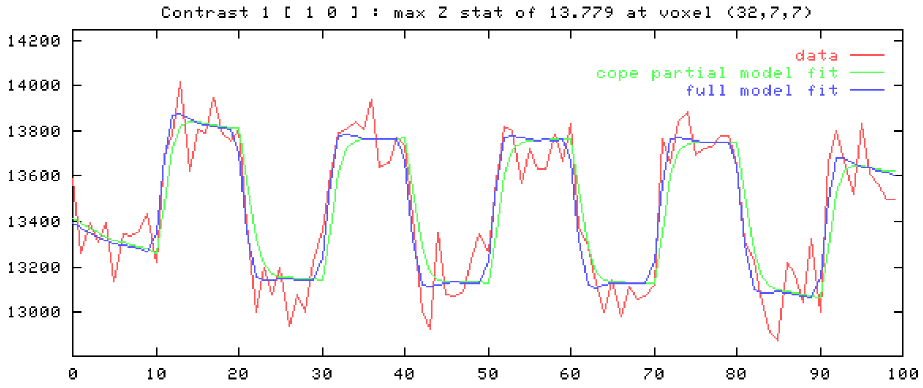
FEAT saves many images to file - various filtered data, statistical output and colour rendered output images - into a separate FEAT directory for each session. If you want to re-run the final stages of analysis ("contrasts, thresholding and rendering"), you can do so without re-running any of the computationally intensive parts of FEAT, by telling FEAT to look in a FEAT directory for all of the raw statistical images which it needs to do this. The results of this re-run of the final stages either overwrite the old ones, or optionally are put in a **new** FEAT directory, named similarly to the original FEAT directory, but with an extra "+" included in the name.

FEAT can also carry out the registration of the low resolution functional images to a high resolution scan, and registration of the high resolution scan to a standard (e.g. Talairached) image. Registration is carried out using FLIRT.

For higher-level analysis (e.g. analysis across sessions or across subjects) FEAT uses FLAME (FMRIB's Local Analysis of Mixed Effects). FLAME uses very sophisticated methods for modelling and estimating the random-effects component of the measured inter-session mixed-effects variance, using MCMC sampling to get an accurate estimation of the true random-effects variance and degrees of freedom at each voxel.

There is a brief overview of GLM analysis in Appendix A and an overview of how the design matrix is setup in FEAT in Appendix B.





Before Running FEAT

Before calling the FEAT GUI, you need to prepare each session's data as a 4D Analyze format image (there are utilities in fsl/bin called avwmerge and avwsplit to convert between multiple 3D images and a single 4D - ie 3D+time - image). If the data requires any scanner-specific corrections (for example, for artefacts such as slice dropouts), this should be applied to the data before running FEAT.

Structural images for use as "highres" images in registration should normally be brain-extracted using BET before being used by FEAT.

FEAT Basics

To call the FEAT GUI, either run **Feat**, or run **fsl** and press the **FEAT** button.

Now set the filename of the 4D input image (e.g. **/users/sibelius/origfunc.hdr**) by pressing **Select 4D data**. You can setup FEAT to process many input images, one after another, as long as they all require exactly the same analysis. Each one will generate its own FEAT directory, the name of which is based on the input data's filename (unless you enter an **Output directory** name).

Note that if you later run **Post-stats** or **Registration**, or if you are running **Higher-level Analysis**, then instead of selecting 4D data as the input, you select FEAT directories. In this case first set the top two drop-down menus in the GUI and then select the FEAT directory or directories; it is important to select the FEAT directories before setting up anything else in FEAT. This is because quite a lot of FEAT settings are loaded from the first selected FEAT directory, possibly over-writing any settings which you wish to change!

If you are running FEAT from within MEDx and the number of **File-based first-level analyses** is set to 0 then FEAT will use the selected group page inside the current MEDx folder for analysis, instead of a file on disk.

Total volumes (including volumes to be deleted) is automatically set from the input files chosen.

Now set **Delete volumes**. These should be the volumes that are not wanted because steady-state imaging is not reached for typically two or three volumes. These volumes are deleted as soon as FEAT is started, so any 4D data output by FEAT will not contain the deleted volumes. Note that **Delete volumes** should not be used to correct for the time lag between stimulation and the measured response - this is corrected for in the design matrix by convolving the input stimulation waveform with a blurring-and-delaying haemodynamic response function. Most importantly, remember when setting up the design matrix that the timings in the design matrix start at t=0 seconds, and this corresponds to the start of the first image taken after the deleted scans. In other words, the design matrix starts **after** the deleted scans have been deleted.

Set the **TR** (time from the start of one volume to the start of the next).

Now set the **High pass filter cutoff** point (seconds), that is, the longest temporal period that you will allow. A sensible setting in the case of an ABAB or ABACABAC type block design is the (A+B) or (A+B+A+C) total cycle time. For event-related designs the rule is not so simple, but in general the cutoff can typically be reduced at least to 50s.

Note that virtually every timing input in FEAT is set in seconds, not volumes. **Total volumes** and **Delete volumes** are exceptions.

Now click on the **Stats** tab and setup the model and required contrasts (for more detail see below).

When FEAT setup is complete and the **Go** button is pressed, the setup gets saved in a temporary FEAT setup file. Then a script (called **feat** - note the lower case) is run which uses the setup file and carries out all the FMRI analysis steps asked for, starting by creating a FEAT results directory, and copying the setup file into here, named **design.fsf** (this setup file can be later loaded back into FEAT using the **Load** button).

Once the script has started running you can **Exit** the FEAT GUI (and even log out of your computer). The analysis will continue until completion, printing text information about its progress in the **Featwatcher** GUI.

FEAT in Detail

First-level or Higher-level Analysis?

Use **First-level analysis** for analysing each session's data - i.e. the time-series analysis of the raw 4D FMRI data.

Use **Higher-level analysis** for combining first-level analyses. You can use this hierarchically - for example at second-level to analyse across several sessions and then at third-level to analyse across several subjects.

Full Analysis or Partial Analysis?

You can run a full analysis - Pre-Stats; Stats; Post-stats; Registration - or a (sensible) subset of these options.

If you select **Post-stats** or **Registration only**, you will need to select a FEAT directory (or directories) instead of starting with 4D image data; the results already produced in those FEAT directories will then be used as appropriate.

Note that if you want to run only **Post-stats**, you must select the FEAT directory/directories before editing the contrasts or thresholding parameters, as these will get reset on selection of the FEAT directory/directories.

Misc

Balloon help (the popup help messages in the FEAT GUI) can be turned off once you are familiar with FEAT.

The **Featwatcher** button allows you to tell **Feat** not to start the graphical program Featwatcher that shows you how the FEAT analysis is progressing. If you are running lots of analyses you probably want to turn this off; you can view the same logging information by looking at the report.log files in any FEAT directories instead, or start the Featwatcher GUI by hand.

For the above two settings, you can control the default behaviour of the FEAT GUI by putting the following, with appropriate values set, in a file called .fsl in your home directory;

```
    set fmri(help_yn) 1
    set fmri(featwatcher_yn) 1
```

If you are using FEAT to carry out multiple analyses, you might want to do this overnight, to reduce the load on your computer. You can use the Delay feature to tell FEAT how long to wait before starting the analyses.

Brain/background threshold, % This is automatically calculated, as a % of the maximum input image intensity. It is used in intensity normalisation, brain mask generation and various other places in the FEAT analysis.

If you are just re-running post-stats or registration, you can either choose to **Overwrite original post-stats results** and registration results, inside the existing FEAT directory, or **Copy original FEAT directory** for a complete copy of the original FEAT directory, with the new results in it.

Data

First set the **Number of analyses**. At first level this is the number of identical analyses you want to carry out. At higher level this is the number of FEAT directories to be input from the lower-level analysis to the higher.

Next set the filename of the 4D input image (e.g. /home/sibelius/func.hdr). You can setup FEAT to process many input images, one after another, as long as they all require exactly the same analysis. Each one will generate its own FEAT directory, the name of which is based on the input data's filename.

Alternatively, if you are running either just **Post-stats** or **Registration only**, or running **Higher-level analysis**, the selection of 4D data changes to the selection of FEAT directories. Note that in this case you should select the FEAT directories before setting up anything else in FEAT (such as changing the thresholds). This is because quite a lot of FEAT settings are loaded from the first selected FEAT directory, possibly over-writing any settings which you wish to change!

If the **Output directory** is left blank, the output FEAT directory name is derived from the input data name. (For higher-level analysis, the output name is derived from the first lower-level FEAT directory selected as input.) If, however, you wish to explicitly choose the output FEAT directory name, for example, so that you can include in the name a hint about the particular analysis that was carried out, you can set this here. This output directory naming behaviour is modified if you are setting up multiple first-level analyses, where you are selecting multiple input data sets and will end up with multiple output FEAT directories. In this case, whatever you enter here will be used and appended to what would have been the default output directory name if you had entered nothing. For example, if you are setting up 3 analyses with input data names **/home/neo/fmri1.hdr**, **/home/neo/fmri2.hdr** and **/home/neo/fmri3.hdr**, and set the output name to **analysisA**, the output directories will end up as **/home/neo/fmri1_analysisA.feat** etc.

Total volumes shows the number of FMRI volumes in the time series, including any initial volumes that you wish to delete. This will get set automatically once valid input data has been selected. The reason for allowing you to set this number by hand before selecting data is so that you can setup and view a model without having any data, for experimental planning purposes etc.

Delete volumes controls the number of initial FMRI volumes to delete before any further processing. You should have decided on this number when the scans were acquired. Typically your experiment would have begun after these initial scans (sometimes called "dummy scans"). These should be the volumes that are not wanted because steady-state imaging has not yet been reached - typically two or three volumes. These volumes are deleted as soon as FEAT is started, so all 4D data sets produced by FEAT will not contain the deleted volumes. Note that **Delete volumes** should not be used to correct for the time lag between stimulation and the measured response - this is corrected for in the design matrix by convolving the input stimulation waveform with a blurring-and-delaying haemodynamic response function. Most importantly, remember when setting up the design matrix, that the timings in the design matrix start at t=0 seconds, and this corresponds to the start of the first image taken after the deleted scans. In other words, the design matrix starts AFTER the **deleted scans** have been deleted.

TR controls the time (in seconds) between scanning successive FMRI volumes.

The **High pass filter cutoff** controls the longest temporal period that you will allow. A sensible setting in the case of an ABAB or ABACABAC type block design is the (A+B) or (A+B+A+C) total cycle time. For event-related designs the rule is not so simple, but in general the cutoff can typically be reduced at least to 50s. This value is setup here rather than in **Pre-stats** because it also affects the generation of the model; the same high pass filtering is applied to the model as to the data, to get the best possible match between the model and data.

Pre-Stats

Slice timing correction corrects each voxel's time-series for the fact that later processing assumes that all slices were acquired exactly half-way through the relevant volume's acquisition time (TR), whereas in fact each slice is taken at slightly different times. Slice timing correction works by using (Hanning-windowed) sinc interpolation to shift each time-series by an appropriate fraction of a TR relative to the middle of the TR period. It is necessary to know in what order the slices were acquired and set the appropriate option here. If slices were acquired from the bottom of the brain to the top select **Regular up**. If slices were acquired from the top of the brain to the bottom select **Regular down**. If the slices were acquired with interleaved order (0, 2, 4 ... 1, 3, 5 ...) then choose the **Interleaved** option. If slices were not acquired in regular order you will need to use a slice order file or a slice timings file. If a slice order file is to be used, create a text file with a single number on each line, where the first line states which slice was acquired first, the second line states which slice was acquired second, etc. The first slice is numbered 1 not 0. If a slice timings file is to be used, put one value (ie for each slice) on each line of a text file. The units are in TRs, with 0.5 corresponding to no shift. Therefore a sensible range of values will be between 0 and 1.

You will normally want to apply **Motion correction**; this attempts to remove the effect of subject head motion during the experiment. MCFLIRT uses FLIRT (FMRIB's Linear Registration Tool) tuned to the problem of FMRI motion correction, applying rigid-body transformations. Note that there is no "spin history" (aka "correction for movement") option with MCFLIRT. This is because this is still a poorly understood correction method which is under further investigation.

By default **BET brain extraction** is applied to create a brain mask from the first volume in the FMRI data. This is normally better than simple intensity-based thresholding for getting rid of unwanted voxels in FMRI data. Note that here, BET is setup to run in a quite liberal way so that there is very little danger of removing valid brain voxels. If the field-of-view of the image (in any direction) is less than 30mm then BET is turned off by default.

Spatial smoothing is carried out on each volume of the FMRI data set separately. This is intended to reduce noise without reducing valid activation; this is successful as long as the underlying activation area is larger than the extent of the smoothing. Thus if you are looking for very small activation areas then you should maybe reduce smoothing from the default of 5mm, and if you are looking for larger areas, you can increase it, maybe to 10 or even 15mm. To turn off spatial smoothing simply set FWHM to 0.

Intensity normalisation forces every FMRI volume to have the same mean intensity. For each volume it calculates the mean intensity and then scales the intensity across the whole volume so that the global mean becomes a preset constant. This step is normally discouraged - hence is turned off by default. When this step is not carried out, the whole 4D data set is still normalised by a single scaling factor ("grand mean scaling") - each volume is scaled by the same amount. This is so that higher-level analyses are valid.

Highpass temporal filtering uses a local fit of a straight line (Gaussian-weighted within the line to give a smooth response) to remove low frequency artefacts. This is preferable to sharp rolloff FIR-based filtering as it does not introduce autocorrelations into the data. By default the same filtering will also be applied to the model.

Stats (First-level)

FILM General Linear Model

General linear modelling allows you to describe one or more stimulation types, and, for each voxel, a linear combination of the modelled response to these stimulation types is found which minimises the unmodelled noise in the fitting. If you are not familiar with the concepts of the GLM and contrasts of parameter estimates, then you should now read Appendix A.

For normal first-level time series analysis you should **Use FILM prewhitening** to make the statistics valid and maximally efficient. For other data - for example, very long TR (>30s) FMRI data, PET data or data with very few time points (<50) - this should be turned off.

You can setup FILM easily for simple designs by using the FILM "wizard" - press the **Simple model setup** button. Then choose whether to setup **ABAB...** or **ABACABAC...** designs (block or single-event). The A blocks will normally be rest (or control) conditions. Enter the timings (in seconds) for these periods and press **Process**; FILM will be automatically setup for you.

If you want to setup a more complex model, or adjust the setup created by the wizard, press **Full model setup** button. This is now described in detail.

EVs

First set the **Number of original EVs** (explanatory variables) - basic number of explanatory variables in the design matrix; this means the number of different effects that you wish to model - one for each modelled stimulation type, and one for each modelled confound. For first-level analyses, it is common for the final design matrix to have a greater number of **real EVs** than this **original** number; for example, when using basis functions, each **original EV** gives rise to several **real EVs**.

Now you need to setup each EV separately. Choose the basic shape of the waveform that describes the stimulus or confound that you wish to model. The basic waveform should be exactly in time with the applied stimulation, i.e., not lagged at all. This is because the measured (time-series) response will be delayed with respect to the stimulation, and this delay is modelled in the design matrix by convolution of the basic waveform with a suitable haemodynamic response function (see below).

For an on/off (or a regularly-spaced single-event) experiment choose a **square** wave. To model single-event experiments with this method, the **On** periods will probably be small - e.g., 1s or even less.

For sinusoidal modelling choose the **Sinusoid** option and select the number of **Harmonics** (or overtones) that you want to add to the fundamental frequency.

For a single-event experiment with irregular timing for the stimulations, a custom file can be used. With **Custom (1 entry per volume)**, you specify a single value for each timepoint. The custom file should be a raw text file, and should be a list of numbers, separated by spaces or newlines, with one number for each volume (after subtracting the number of deleted images). These numbers can either all be 0s and 1s, or can take a range of values. The former case would be appropriate if the same stimulus was applied at varying time points; the latter would be appropriate, for example, if recorded subject responses are to be inserted as an effect to be modelled. Note that it may or may not be appropriate to convolve this particular waveform with an HRF - in the case of single-event, it is.

For even finer control over the input waveform, choose **Custom (3 column format)**. In this case the custom file consists of triplets of numbers; you can have any number of triplets. Each triplet describes a short period of time and the value of the model during that time. The first number in each triplet is the onset (in seconds) of the period, the second number is the duration (in seconds) of the period, and the third number is the value of the input during that period. The same comments as above apply, about whether these numbers are 0s and 1s, or vary continuously. The start of the first non-deleted volume correpsonds to t=0.

Note that whilst ALL columns are demeaned before model fitting, neither custom format will get rescaled - it is up to you to make sure that relative scaling between different EVs is sensible. If you double the scaling of values in an EV you will halve the resulting parameter estimate, which will change contrasts of this EV against others.

If you select **Interaction** then you can set which EVs this EV models the interaction between. This EV is produced by multiplying together selected EVs, and allows the modelling of the non-additive interaction between the selected EVs. This requires that the EVs are sometimes on at the same time, and sometimes on separately.

If you have chosen a **Square** or **Sinusoid** basic shape, you then need to specify what the timings of this shape are. **Skip** is the initial period of zeros (in seconds) before the waveform commences. **Off** is the duration (seconds) of the "Off" periods in the square wave. **On** is the duration (seconds) of the "On" periods in the square wave. **Period** is the period (seconds) of the **Sinusoid** waveform. **Phase** is the phase shift (seconds) of the waveform; by default, after the **Skip** period, the square wave starts with a full **Off** period and the **Sinusoid** starts by falling from zero. However, the wave can be brought forward in time according to the phase shift. Thus to start with half of a normal **Off** period, enter the **Phase** as half of the **Off** period. To start with a full **On** period, enter the same as the **Off** period. **Stop after** is the total duration (seconds) of the waveform, starting after the **Skip** period. "-1" means do not stop. After stopping a waveform, all remaining values in the model are set to zero.

Convolution sets the form of the HRF (haemodynamic response function) convolution that will be applied to the basic waveform. This blurs and delays the original waveform, in an attempt to match the difference between the input function (original waveform, i.e., stimulus waveform) and the output function (measured FMRI haemodynamic response). If the original waveform is already in an appropriate form, e.g., was sampled from the data itself, **None** should be selected. The next three options are all somewhat similar blurring and delaying functions. **Gaussian** is simply a Gaussian kernel, whose width and lag can be altered. **Gamma** is a Gamma variate (in fact a normalisation of the probability density function of the Gamma function); again, width and lag can be altered. **Double-Gamma HRF** is a preset function which is a mixture of two Gamma functions - a standard positive function at normal lag, and a small, delayed, inverted Gamma, which attempts to model the late undershoot.

The remaining convolution options setup different **basis functions**. This means that the original EV waveform will get convolved by a **basis set** of related but different convolution kernels - a set of Gammas of different widths and lags, a set of sinusoids of differing frequencies or a set of finite-impulse-response (FIR) filters (with FIR the convolution kernel is represented as a set of discrete fixed-width **impulses**). Therefore the **original EV** will generate a set of **real EVs**, one for each basis function.

You should normally apply the same temporal filtering to the model as you have applied to the data, as the model is designed to look like the data before temporal filtering was applied. In this way, long-time-scale components in the model will be dealt with correctly. This is set with the **Apply temporal filtering** option.

Adding a fraction of the temporal derivative of the blurred original waveform is equivalent to shifting the waveform slightly in time, in order to achieve a slightly better fit to the data. Thus adding in the temporal derivative of a waveform into the design matrix allows a better fit for the whole model, reducing unexplained noise, and increasing resulting statistical significances. Thus, setting **Add temporal derivative** produces a new waveform in the final design matrix (next to the waveform from which it was derived) This option is not available if you are using basis functions.

Orthogonalising an EV with respect to other EVs means that it is completely independent of the other EVs, i.e. contains no component related to them. Most sensible designs are already in this form - all EVs are at least close to being orthogonal to all others. However, this may not be the case; you can use this facility to force an EV to be orthogonal to some or all other EVs. This is achieved by subtracting from the EV that part which is related to the other EVs selected here. An example use would be if you had another EV which was a constant height spike train, and the current EV is derived from this other one, but with a linear increase in spike height imposed, to model an increase in response during the experiment for any reason. You would not want the current EV to contain any component of the constant height EV, so you would orthogonalise the current EV wrt the other.

Contrasts

Each EV (explanatory variable, i.e., waveform) in the design matrix results in a PE (parameter estimate) image. This estimate tells you how strongly that waveform fits the data at each voxel - the higher it is, the better the fit. For an unblurred square wave input (which will be scaled in the model from -0.5 to 0.5), the PE image is equivalent to the "mean difference image". To convert from a PE to a t statistic image, the PE is divided by its standard error, which is derived from the residual noise after the complete model has been fit. The t image is then transformed into a Z statistic via standard statistical transformation. As well as Z images arising from single EVs, it is possible to combine different EVs (waveforms) - for example, to see where one has a bigger effect than another. To do this, one PE is subtracted from another, a combined standard error is calculated, and a new Z image is created.

All of the above is controlled by you, by setting up contrasts. Each output Z statistic image is generated by setting up a contrast vector; thus set the number of outputs that you want, using **Number of contrasts**. To convert a single EV into a Z statistic image, set its contrast value to 1 and all others to 0. Thus the simplest design, with one EV only, has just one contrast vector, and only one entry in this contrast vector; 1. To add more contrast vectors, increase the **Number of contrasts**. To compare two EVs, for example, to subtract one stimulus type (EV1) from another type (EV2), set EV1's contrast value to -1 and EV2's to 1. A Z statistic image will be generated according to this request.

For first-level analyses, it is common for the final design matrix to have a greater number of **real EVs** than the **original** number; for example, when using basis functions, each **original EV** gives rise to several **real EVs**. Therefore it is possible in many cases for you to setup contrasts and F-tests with respect to the **original EVs**, and FEAT will work out for you what these will be for the final design matrix. For example, a single [1] contrast on an original EV for which basis function HRF convolutions have been chosen will result in a single [1] contrast for each resulting real EV, and then an F-test across these. In general you can switch between setting up contrasts and F-tests with respect to **Original EVs** and **Real EVs**; though of course if you fine-tune the contrasts for real EVs and then revert to original EV setup some settings may be lost. When you **View** the design matrix or press **Done** at the end of setting up the model, an **Original EVs** setup will get converted to the appropriate **Real EVs** settings.

An important point to note is that you should **not** test for differences between different conditions (or at higher-level, between sessions) by looking for differences between their separate individual analyses. One could be just above threshold and the other just below, and their difference might not be significant. The correct way to tell whether two conditions or session's analyses are significantly different is to run a differential contrast like [1 -1] between them (or, at higher-level, run a higher-level FEAT analysis to contrast lower-level analyses); this contrast will then get properly thresholded to test for significance.

There is another important point to note when interpreting differential (eg [1 -1]) contrasts. This is that you are quite likely to only want to check for A>B if both are positive. Don't forget that if both A and B are negative then this contrast could still come out significantly positive! In this case, the thing to do is to use the **Contrast masking** feature (see below); setup contrasts for the individual EVs and then mask the differential contrast with these.

F-tests

F-tests enable you to investigate several contrasts at the same time, for example to see whether any of them (or any combination of them) is significantly non-zero. Also, the F-test allows you to compare the contribution of each contrast to the model and decide on significant and non-significant ones. F-tests are non-directional (i.e. test for "positive" and "negative" activation).

One example of F-test usage is if a particular stimulation is to be represented by several EVs, each with the same input function (e.g. square wave or custom timing) but all with different HRF convolutions - i.e. several **basis functions**. Putting all relevant resulting parameter estimates together into an F-test allows the complete fit to be tested against zero without having to specify the relative weights of the basis functions (as one would need to do with a single contrast). So - if you had three basis functions (EVs 1,2 and 3) the wrong way of combining them is a single (T-test) contrast of [1 1 1]. The right way is to make three contrasts [1 0 0] [0 1 0] and [0 0 1] and enter all three contrasts into an F-test. As described above, FEAT will automatically do this for you if you set up contrasts for **original EVs** instead of **real EVs**.

You can carry out as many F-tests as you like. Each test includes the particular contrasts that you specify by clicking on the appropriate buttons.

Buttons

To view the current state of the design matrix, press **View design**. This is a graphical representation of the design matrix and parameter contrasts. The bar on the left is a representation of time, which starts at the top and points downwards. The white marks show the position of every 10th volume in time. The red bar shows the period of the longest temporal cycle which was passed by the highpass filtering. The main top part shows the design matrix; time is represented on the vertical axis and each column is a different (real) explanatory variable (e.g., stimulus type). Both the red lines and the black-white images represent the same thing - the variation of the waveform in time. Below this is shown the requested contrasts; each row is a different contrast vector and each column refers to the weighting of the relevant explanatory variable. Thus each row will result in a Z statistic image. If F-tests have been specified, these appear to the right of the contrasts; each column is a different F-test, with the inclusion of particular contrasts depicted by filled squares instead of empty ones.

If you have more than one EV and you press **Covariance** you will see a graphical representation of the covariance matrix of the design matrix. The first matrix shows the absolute value of the normalised correlation of each EV with each EV. If a design is well-conditioned (i.e. not approaching rank deficiency) then the diagonal elements should be white and all others darker. So - if there are any very bright elements off the diagonal, you can immediately tell which EVs are too similar to each other - for example, if element [1,3] (and [3,1]) is bright then columns 1 and 3 in the design matrix are possibly too similar. Note that this includes all real EVs, including any added temporal derivatives, basis functions, etc. The second matrix shows a similar thing after the design matrix has been run through SVD (singular value decomposition). All non-diagonal elements will be zero and the diagonal elements are given by the eigenvalues of the SVD, so that a poorly-conditioned design is obvious if any of the diagonal elements are black.

When you have finished setting up the design matrix, press **Done**. This will dismiss the GLM GUI, and will give you a final view of the design matrix.

Contrasts, Thresholding, Rendering

If you are not carrying out a **Full analysis** and are re-running **Post-stats**, a button appears to allow you to **Edit contrasts**. This allows setup of contrasts and F-tests, to be run on the previous analysis.

If you choose a mask for **Pre-threshold masking** then all stats images will be masked by the chosen mask before thresholding. There are two reasons why you might want to do this. The first is that you might want to constrain your search for activation to a particular area. The second is that in doing so, you are reducing the number of voxels tested and therefore will make any multiple-comparison-correction in the thresholding less stringent. The mask image chosen does not have to be a binary mask - for example, it can be a thresholded stats image from a previous analysis (in the same space as the data to be analysed here); only voxels containing zero in the **mask** image will get zeroed in this masking process. If pre-threshold masking is used, it is still necessary to carry out thresholding.

Thresholding: After carrying out the initial statistical test, the resulting Z statistic image is then normally thresholded to show which voxels or clusters of voxels are activated at a particular significance level.

If **Cluster** thresholding is selected, a Z statistic threshold is used to define contiguous clusters. Then each cluster's estimated significance level (from GRF-theory) is compared with the cluster probability threshold. Significant clusters are then used to mask the original Z statistic image for later production of colour blobs. This method of thresholding is an alternative to **Voxel**-based correction, and is normally more sensitive to activation. You may well want to increase the cluster creation **Z threshold** if you have high levels of activation.

The FEAT web page report includes a table of cluster details, viewed by clicking on the relevant colour-overlay image. Note that cluster p-values are not given for contrasts where post-threshold contrast masking (see below) is applied, as there is not a sensible p-value associated with the new clusters formed after masking.

If **Voxel** thresholding is selected, GRF-theory-based maximum height thresholding is carried out, with thresholding at the level set, using one-tailed testing. This test is less overly-conservative than Bonferroni correction.

You can also choose to simply threshold the uncorrected Z statistic values, or apply no thresholding at all.

Contrast masking: You can setup the masking of contrasts by other contrasts; after thresholding of all contrasts has taken place you can further **threshold** a given Z statistic image by masking it with non-zeroed voxels from other contrasts.

This means that of the voxels which passed thresholding in the contrast (or F-test) of interest, only those which also survived thresholding in the other contrasts (or F-tests) are kept.

As a further option, the generated masks can be derived from all positive Z statistic voxels in the mask contrasts rather than all voxels that survived thresholding.

Rendering: The Z statistic range selected for rendering is automatically calculated by default, to run from red (minimum Z statistic after thresholding) to yellow (maximum Z statistic). If more than one colour rendered image is to be produced (i.e., when multiple constrasts are created) then the overall range of Z values is automatically found from all of the Z statistic images, for consistent Z statistic colour-coding.

If multiple analyses are to be carried out separately, **Use preset Z min/max** should be chosen, and the min/max values set by hand. Again, this ensures consistency of Z statistic colour-coding - if several experiments are to be reported side-by-side, colours will refer to the same Z statistic values in each picture. When using this option, you should choose a conservatively wide range for the min and max (e.g., min=1, max=15), to make sure that you do not carry out unintentional thresholding via colour rendering.

With **Solid colours** you don't see any sign of the background images within the colour blobs; with **Transparent colours** you will see through the colour blobs to the background intensity.

If you are running a **Higher-level analysis** you can select what image will be used as the background image for the activation colour overlays. The default of

Mean highres is probably the best for relating activation to underlying structure. For a sharper underlying image, (but one which is not so representative of the group of subjects), you can instead choose to use the highres image from the first selected subject. You can alternatively choose to use the original lowres functional data for the overlays, or the standard-space template image.

Registration

Before any multi-session or multi-subject analyses can be carried out, the different sessions need to be registered to each other. This is made easy with FEAT, by saving the appropriate transformations inside the FEAT directories; the transformations are then applied when group statistics is carried out, to transform any relevant statistic images into the common space. By doing this (saving the relevant registration transformations and only applying them to the stats images later) a lot of disk space is saved.

Registration inside FEAT uses FLIRT (FMRIB's Linear Image Registration Tool). This is a very robust affine registration program which can register similar type images (intra-modal) or different types (inter-modal).

Typically, registration in FEAT is a two-stage process. First an example FMRI low resolution image is registered to an example high resolution image (often the same subject's T1-weighted structural). The transformation for this is saved into the FEAT directory. Then the high res image is registered to a standard image (normally a T1-weighted image in standard space, such as the MNI 152 average image). This transformation, also, is saved. Finally, the two transformations are combined into a third, which will take the low resolution FMRI images (and the statistic images derived from the first-level analyses) straight into standard space, when applied later, during group analysis.

You can carry out registration for each first-level analysis at the same time as the original analysis, or get FEAT to "register" a pre-existing FEAT directory, at a later time. In the latter case, change the **Full analysis** to **Registration only**.

The **Initial structural image** is the high resolution structural image which the low resolution functional data will be registered to, and this in turn will be registered to the main highres image. It only makes sense to have this initial highres image if a main highres image is also specified and used in the registration.

One example of an initial highres structural image might be a medium-quality structural scan taken during a day's scanning, if a higher-quality image has been previously taken for the subject. A second example might be a full-brain image with the same MR sequence as the functional data, useful if the actual functional data is only partial-brain. It is strongly recommended that this image have non-brain structures already removed, for example by using BET.

If the field-of-view of the functional data (in any direction) is less than 120mm, then the registration of the functional data will by default have a reduced degree-of-freedom, for registration stability.

If you are attempting to register partial field-of-view functional data to a whole-brain image then **3 DOF** is recommended - in this case only translations are allowed.

If the orientation of any image is different from any other image it may be necessary to change the search to **Full search**.

The **Main structural image** is is the main high resolution structural image which the low resolution functional data will be registered to (optionally via the **initial structural image**), and this in turn will be registered to the standard brain. It is strongly recommended that this image have non-brain structures already removed, for example by using BET.

Standard space refers to the standard (reference) image; it should be an image already in Talairach space, ideally with the non-brain structures already removed.

Bottom Row of Buttons

When you have finished setting up FEAT, press **Go** to run the analysis. Once FEAT is running, you can either **Exit** the GUI, or setup further analyses.

The **Save** and **Load** buttons enable you to save and load the complete FEAT setup to and from file. The filename should normally be chosen as design.fsf - this is also the name which FEAT uses to automatically save the setup inside a FEAT directory. Thus you can load in the setup that was used on old analyses by loading in this file from old FEAT directories.

The **Utils** button produces a menu of FEAT-related utilities:

- **Load FEAT results into MEDx** (only seen if FEAT is run from within MEDx). This opens a new folder and loads in results from a FEAT directory, setting up each stats image to be "time series clickable".
- Featquery is a program which allows you to interrogate FEAT results by defining a mask or set of co-ordinates (in standard-space, highres-space or loweres-space) and get mean stats values and time-series.
- Simple stats colour rendering allows you to overlay one or two stats images on a background image of the same size.
- **Colour render FEAT stats in high res** produces colour rendered stats images in a selected FEAT directory, using either the original high

resolution structural image as the background, or the structural image transformed into Talairach space as the background. This script starts by transforming stats into high resolution space and then produces 3D colour overlay images. 2D pictures of these are then saved to file and the 3D files removed to save disk space, but this removal can be turned off in the GUI.

Time-Series Plots

FEAT generates a set of time-series plots for data vs model for peak Z voxels. The main FEAT report web page contains a single plot for each contrast (from the peak voxel); clicking on this takes you to more plots related to that contrast, including also, in the case of cluster-based thresholding, plots averaged over all significant voxels.

Plots of **full model fit** vs **data** show the original data and the complete model fit given by the GLM analysis.

Plots of **cope partial model fit** vs **reduced data** show the model fit due simply to the contrast of interest versus that part of the data which is relevant to the reduced model (i.e. full data minus full model plus cope partial model). This generally is only easily interpretable in the case of simple non-differential contrasts.

Peri-stimulus plots show the same plots as described above but averaged over all "repeats" of events, whether ON-OFF blocks in a block-design, or events in an event-related design. Thus you get to see the average response shape. Note that FEAT tries to guess what an "event" is in your design automatically, so in complex designs this can give somewhat strange looking plots! The peri-stimulus plots are for the peak voxel only; one pair of full/partial plots is produced for each EV in the design matrix for that peak voxel, with the "events" defined from that EV only.

Group Statistics

Background and Theory

For higher-level analysis (e.g. analysis across sessions or across subjects) FEAT uses FLAME (FMRIB's Local Analysis of Mixed Effects). FLAME uses sophisticated methods for modelling (see related techreport TR01CB1) and estimating the inter-session or inter-subject random-effects component of the mixed-effects variance, by using MCMC to get an accurate estimation of the true random-effects variance and degrees of freedom at each voxel.

"Mixed-effects" (ME) variance is the sum of "fixed-effects" (FE) variance (the within-session across-time variances estimated in the first-level analyses) and "random-effects" (RE) variance (the "true" cross-session variances of first-level parameter estimates). Note that the labels "mixed effects" and "random effects" are often (incorrectly) used interchangeably, partly because they are in practice often (but, importantly, not always) quite similar.

Thus, when referring to FEAT-based group analysis, it should be described as mixed-effects, although it may avoid confusion to also add explanatory parenthesis, for example: "mixed-effects (often referred to as 'random-effects') group analysis was carried out".

One factor that makes FEAT's approach to higher-level modelling particularly powerful is that it is easy to model and estimate different variances for different groups in the model. For example, an unpaired two-group comparison (e.g. between controls and patients) can be analysed with separate estimates of variance for each group. It is simply a case of specifying in the GUI what group each subject belongs to. (Note - FLAME does not model different group variances differently in the case of higher-level F-tests, due to the complexity of the resulting distributions; this may be addressed in the future.)

A second sophistication not normally available in multi-level analyses is the carrying-up of the first-level (FE) variances to the higher-level analyses. This means that the FE component of the higher-level ME variance can be taken into account when attempting to estimate the ME variance. One reason why it is suboptimal to simply use the directly-estimated ME variance is that this is often in practice lower than the estimated FE variance, a logical impossibility which implies negative RE variance. FEAT forces the RE variance in effect to be non-negative, giving a better estimate of ME variance.

Another reason for wanting to carry up first-level variances to the higher-level analyses is that it is not then necessary for first-level design matrices to be identical (ie "balanced designs" - for example having the same number of time points or event timings). (Note though: the "height" of design matrix waveforms at first-level must still be compatible across analyses.)

A third advantage in higher-level analysis with FEAT is that it is not necessary for different groups to have the same number of subjects (another aspect to design balancing) for the statistics to be valid, because of the ability to model different variances in different groups.

The higher-level estimation method in FEAT (FLAME) uses the above modelling theory and estimates the higher-level parameter estimates and ME variance using sophisticated estimation techniques. First, the higher-level model is fit using a fast approximation to the final estimation ("FLAME stage 1"). Then, all voxels which are close to threshold (according to the selected contrasts and thresholds) are processed with a much more sophisticated estimation process involving implicit estimation of the ME variance, using MH MCMC (Metropolis-Hastings Markov Chain Monte Carlo sampling) to give the distribution for higher-level contrasts of parameter estimates, to which a general t-distribution is then fit. Hypothesis testing can then be carried out on the fitted t-distribution to give inference based upon the best implicit estimation of the ME variance.

Setting Up Higher-Level Analysis in FEAT

For figures showing the file and directory structures for first- and second-level FEAT analyses, see the *Output section*.

First change **First-level analysis** to **Higher-level** analysis. Note that the only processing option available is **Stats + Post-stats**; this is because at higher-level both **Stats** and **Post-stats** always need setting up, as the thresholding to be carried out affects the functioning of the core stats estimation. For the same reason it is not possible to re-threshold a higher-level analysis - the whole higher-level analysis must be re-run.

You can choose whether your higher-level design matrix will be applied to a set of **lower-level cope images** or a set of **lower-level FEAT directories**. In the latter, more normal, case, each contrast in the lower-level FEAT directories will have the higher-level model applied, each resulting in its own FEAT directory within the new group FEAT directory.

Now set the **Number of analyses** and **Select FEAT directories** (the first-level FEAT analyses to be fed into this higher-level analysis). FEAT will produce a new directory containing the group stats results; unless you specify an output directory name, the output directory name will be derived from the name of the first selected first-level FEAT directory. The suffix .gfeat is used.

Now setup the **Stats**. OLS (ordinary least squares) is a fast estimation technique which ignores all lower-level variance estimation and applies a very simple higher-level model. For the most accurate estimation of higher-level activation you should use FLAME (FMRIB's Local Analysis of Mixed Effects) modelling and estimation. This is a sophisticated two-stage process using Bayesian modelling and estimation (for example it allows separate modelling of the variance in different subject groups, and forces random effects variance to be non-negative). The first stage of FLAME is significantly more accurate than OLS, and nearly as fast. The second stage of FLAME increases accuracy slightly over the first stage, but is quite a lot slower (typically 45-200 minutes). Therefore we recommend that you use the default **FLAME** setting unless you are in a hurry, in which case we recommend using the **FLAME stage 1** only option.

Next go to the full model setup. First choose the **Number of EVs** (different effects to be modelled). Next, the **Number of groups** is the number of different groups (of lower-level sessions or subjects). If you ask for more than one group, each group will end up with a separate estimate of variance for the higher-level parameter estimates; for example, if the first 10 inputs are first-level FEAT outputs from control subjects and the next 10 inputs are first-level FEAT outputs from patients, you can setup two different groups and each will end up with its own variance estimates, possibly improving the final modelling and estimation quality (see examples below for further clarification).

Unlike with first-level analyses, the data (and the model) does **not** get demeaned. This is because mean effects are usually of interest! One effect of this is that a two-group unpaired model needs 2 EVs - one for finding each group's mean; it will not work to have a single EV containing 1's and -1's.

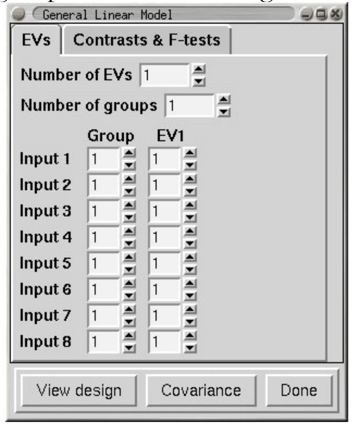
Now setup the required **Contrasts & F-tests** and **Post-stats** (see examples below).

The higher-level design matrix is applied separately to each of the lower-level contrasts; thus each lower-level contrast will result in a new FEAT directory within the new top-level group FEAT directory. When FEAT has completed the higher-level analyses the new top-level group FEAT output directory contains a web page report which contains: a link to each of the original lower-level FEAT directories; a link to each of the higher-level FEAT analyses (one for each lower-level contrast); the picture of the higher-level design matrix.

We now give specific examples of how to set up the most common high-level analyses.

Single-Group Average (One-Sample T-Test)

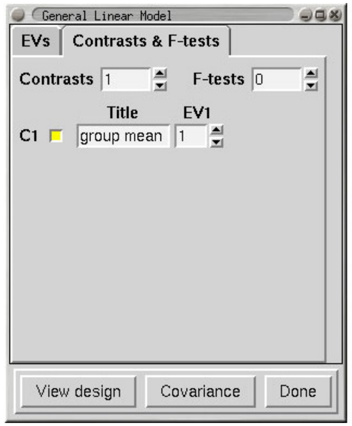
We have 8 subjects all in one group and want the mean group effect. Does the group activate on average?



Number of EVs 1

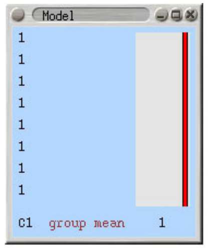
Number of groups 1

	Group	EV1
Input 1	1	1
Input 2	1	1
Input 3	1	1
Input 4	1	1
Input 5	1	1
Input 6	1	1
Input 7	1	1
Input 8	1	1



Contrasts 1 F-tests 0

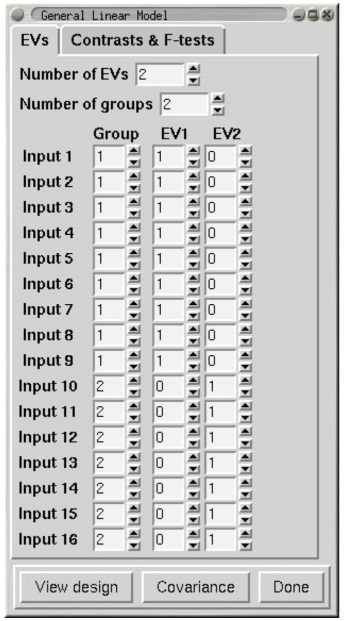
	Title	EV1
C1	group mean	1



```
1
1
1
1
1
1
1
1
C1  group mean    1
```

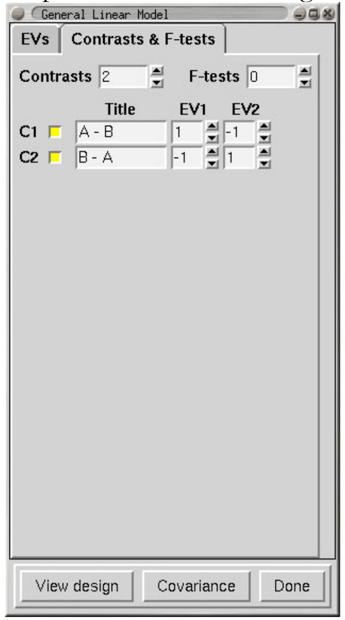
Unpaired Two-Group Difference (Two-Sample Unpaired T-Test)

We have two groups of different kinds of subjects (eg 9 patients and 7 controls) with potentially different cross-subject variance, so we will specifiy two group "memberships" so that FEAT estimates each group's variance separately. We want the mean group difference, and will look in both directions, hence the two contrasts. Note that we cannot setup this model with a single EV (see above).



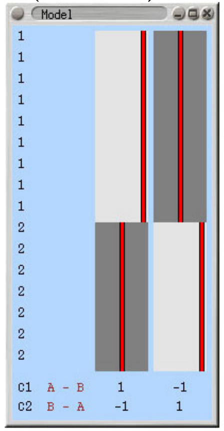
Number of EVs 2
Number of groups 2

	Group	EV1	EV2
Input 1	1	1	0
Input 2	1	1	0
Input 3	1	1	0
Input 4	1	1	0
Input 5	1	1	0
Input 6	1	1	0
Input 7	1	1	0
Input 8	1	1	0
Input 9	1	1	0
Input 10	2	0	1
Input 11	2	0	1
Input 12	2	0	1
Input 13	2	0	1
Input 14	2	0	1
Input 15	2	0	1
Input 16	2	0	1



Contrasts 2 F-tests 0

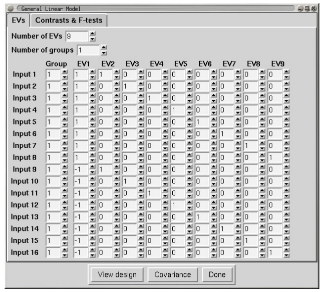
	Title	EV1	EV2
C1	A - B	1	-1
C2	B - A	-1	1



C1	A - B	1	-1
C2	B - A	-1	1

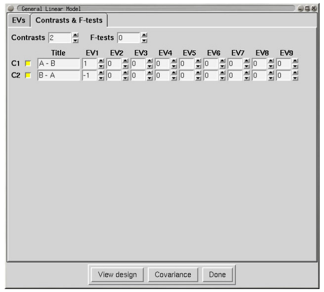
Paired Two-Group Difference (Two-Sample Paired T-Test)

We have a group of 8 subjects scanned under two different conditions, A and B. We enter the condition A analyses as the first 8 inputs, and the condition B analyses as the second 8 inputs. Make sure that the subjects are in the same order within each group of 8! We need one EV for the A-B differences, and then one extra EV for each subject, making 9 in all. EVs 2-9 model each subject's mean effect - in this analysis this is a confound, i.e. parameter estimates 2-9 are ignored, but without this part of the model, the mean effects would intefere with the estimation of the A-B paired differences. A contrast with a one for EV1 and zeros elsewhere tests for A-B paired differences.



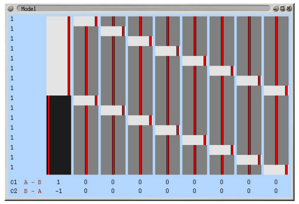
Number of EVs 9
Number of groups 1

	Group	EV1	EV2	EV3	EV4	EV5	EV6	EV7	EV8	EV9
Input 1	1	1	1	0	0	0	0	0	0	0
Input 2	1	1	0	1	0	0	0	0	0	0
Input 3	1	1	0	0	1	0	0	0	0	0
Input 4	1	1	0	0	0	1	0	0	0	0
Input 5	1	1	0	0	0	0	1	0	0	0
Input 6	1	1	0	0	0	0	0	1	0	0
Input 7	1	1	0	0	0	0	0	0	1	0
Input 8	1	1	0	0	0	0	0	0	0	1
Input 9	1	-1	1	0	0	0	0	0	0	0
Input 10	1	-1	0	1	0	0	0	0	0	0
Input 11	1	-1	0	0	1	0	0	0	0	0
Input 12	1	-1	0	0	0	1	0	0	0	0
Input 13	1	-1	0	0	0	0	1	0	0	0
Input 14	1	-1	0	0	0	0	0	1	0	0
Input 15	1	-1	0	0	0	0	0	0	1	0
Input 16	1	-1	0	0	0	0	0	0	0	1



Contrasts 2 F-tests 0

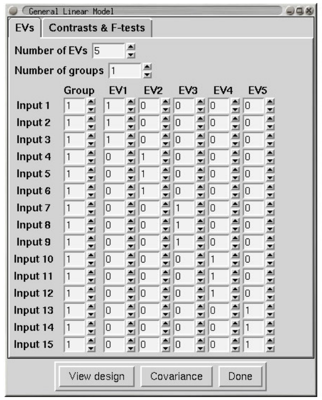
	Title	EV1	EV2	EV3	EV4	EV5	EV6	EV7	EV8	EV9
C1	A - B	1	0	0	0	0	0	0	0	0
C2	B - A	-1	0	0	0	0	0	0	0	0



C1	A - B	1	0	0	0	0	0	0	0	0
C2	B - A	-1	0	0	0	0	0	0	0	0

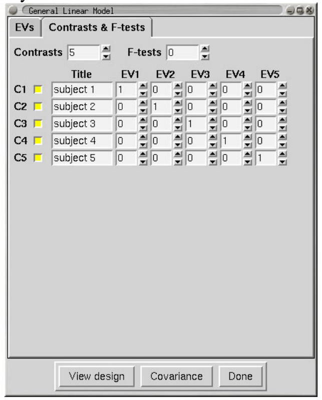
Multi-Session & Multi-Subject (Repeated Measures - Three Level Analysis)

5 subjects each have three sessions. Because the number of sessions is low, we are going to make the estimation of cross-session variance more robust by assuming it is the same for all subjects and putting all subjects into one second-level analysis. Thus for this second-level analysis:



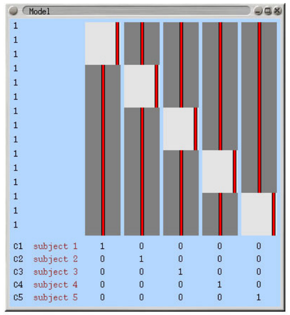
Number of EVs 5
Number of groups 1

	Group	EV1	EV2	EV3	EV4	EV5
Input 1	1	1	0	0	0	0
Input 2	1	1	0	0	0	0
Input 3	1	1	0	0	0	0
Input 4	1	0	1	0	0	0
Input 5	1	0	1	0	0	0
Input 6	1	0	1	0	0	0
Input 7	1	0	0	1	0	0
Input 8	1	0	0	1	0	0
Input 9	1	0	0	1	0	0
Input 10	1	0	0	0	1	0
Input 11	1	0	0	0	1	0
Input 12	1	0	0	0	1	0
Input 13	1	0	0	0	0	1
Input 14	1	0	0	0	0	1
Input 15	1	0	0	0	0	1



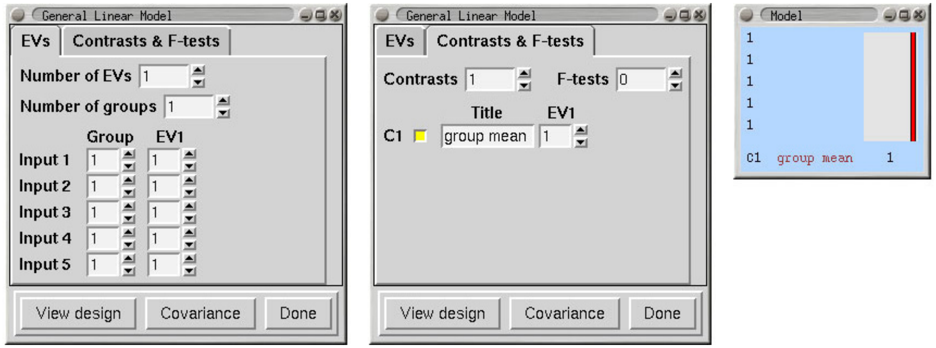
Contrasts 5 F-tests 0

	Title	EV1	EV2	EV3	EV4	EV5
C1	subject 1	1	0	0	0	0
C2	subject 2	0	1	0	0	0
C3	subject 3	0	0	1	0	0
C4	subject 4	0	0	0	1	0
C5	subject 5	0	0	0	0	1



C1	subject 1	1	0	0	0	0
C2	subject 2	0	1	0	0	0
C3	subject 3	0	0	1	0	0
C4	subject 4	0	0	0	1	0
C5	subject 5	0	0	0	0	1

Now we want the mean group effect, across subjects, achieved with a third-level analysis. Select **Inputs are lower-level cope images** and select the 5 cope images created at second-level, found inside the <something>.gfeat/cope1.feat/stats directory.



Number of EVs 1
Number of groups 1

	Group	EV1
Input 1	1	1
Input 2	1	1
Input 3	1	1
Input 4	1	1
Input 5	1	1

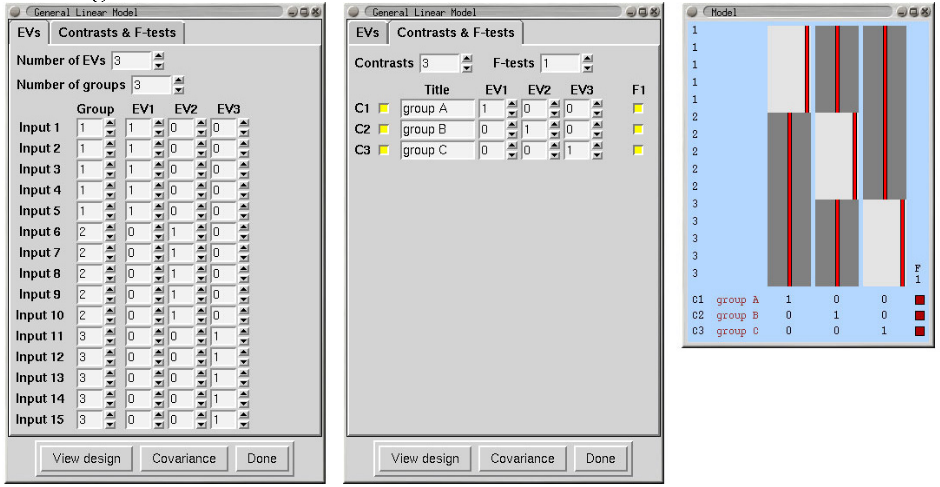
Contrasts 1
F-tests 0

	Title	EV1
C1	group mean	1

1
1
1
1
1
C1 group mean 1

F-Tests

For example, three groups of subjects, with the question - is any group activating on average?



Number of EVs 3
Number of groups 3

	Group	EV1	EV2	EV3
Input 1	1	1	0	0
Input 2	1	1	0	0
Input 3	1	1	0	0
Input 4	1	1	0	0
Input 5	1	1	0	0
Input 6	2	0	1	0
Input 7	2	0	1	0
Input 8	2	0	1	0
Input 9	2	0	1	0
Input 10	2	0	1	0
Input 11	3	0	0	1
Input 12	3	0	0	1
Input 13	3	0	0	1
Input 14	3	0	0	1
Input 15	3	0	0	1

Contrasts 3
F-tests 1

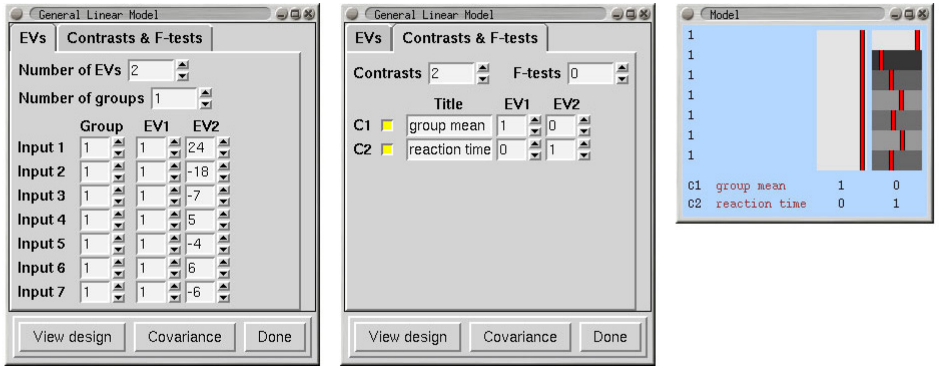
	Title	EV1	EV2	EV3	F1
C1	group A	1	0	0	☑
C2	group B	0	1	0	☑
C3	group C	0	0	1	☑

1
1
1
1
1
2
2
2
2
2
3
3
3
3
3

C1	group A	1	0	0	■
C2	group B	0	1	0	■
C3	group C	0	0	1	■

Single-Group Average with Additional Covariate

We have 7 subjects all in one group. We also have additional measurements such as age, disability scale or behavioural measures such as mean reaction times. The additional effect of the extra measures can be found by entering this as an extra EV which has been orthogonalised wrt the group mean EV - so in this case simply demeaned:



Number of EVs 2
Number of groups 1

	Group	EV1	EV2
Input 1	1	1	24
Input 2	1	1	-18
Input 3	1	1	-7
Input 4	1	1	5
Input 5	1	1	-4
Input 6	1	1	6
Input 7	1	1	-6

Contrasts 2
F-tests 0

	Title	EV1	EV2
C1	group mean	1	0
C2	reaction time	0	1

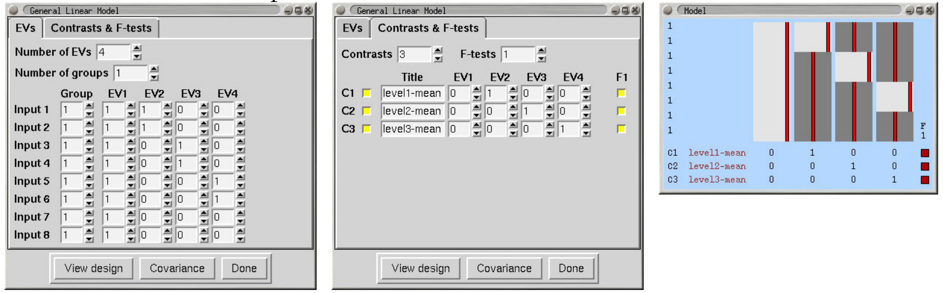
1
1
1
1
1
1
1

C1	group mean	1	0
C2	reaction time	0	1

ANOVA : 1-factor 4-levels

We have 8 subjects and 1 factor at 4 levels. To compare a level with another we could just have one EV per level. However, if we want to ask the ANOVA question - where is there **any** treatment effect then we can do the following. EV1 takes out the global mean (ie across all levels of the factor). EV2 fits cell 1 relative to this mean etc. Thus contrast 1 gives cell 1 (level 1) relative to the global mean etc. A fourth contrast is not necessary as that would add no extra information. The F-test then tests for any deviation from the mean - ie any difference between the levels.

Note - the first two inputs are level 1, the next two are level 2 etc.



Number of EVs 4
Number of groups 1

	Group	EV1	EV2	EV3	EV4
Input 1	1	1	1	0	0
Input 2	1	1	1	0	0
Input 3	1	1	0	1	0
Input 4	1	1	0	1	0
Input 5	1	1	0	0	1
Input 6	1	1	0	0	1
Input 7	1	1	0	0	0
Input 8	1	1	0	0	0

Contrasts 3
F-tests 1

	Title	EV1	EV2	EV3	EV4	F1
C1	level1-mean	0	1	0	0	☑
C2	level2-mean	0	0	1	0	☑
C3	level3-mean	0	0	0	1	☑

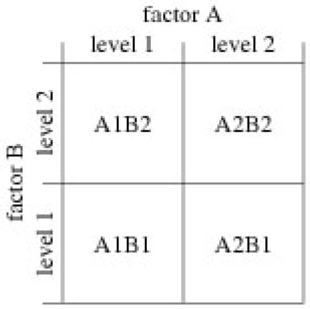
1
1
1
1
1
1
1
1

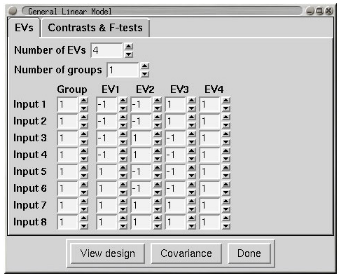
C1	level1-mean	0	1	0	0	■
C2	level2-mean	0	0	1	0	■
C3	level3-mean	0	0	0	1	■

ANOVA : 2-factors 2-levels

Fixed Effects

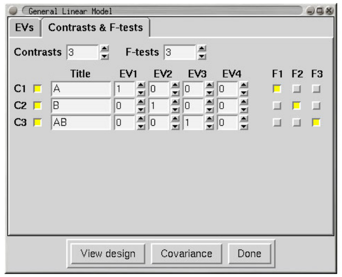
We have 8 subjects and 2 factors, each at 2 levels. To carry out a standard ANOVA we use the following, the three F-tests giving the standard ANOVA results for factor A, factor B and the interaction effect. This assumes that both factors are fixed effects.





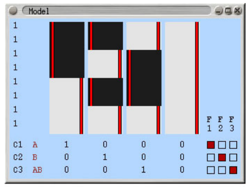
Number of EVs 4
Number of groups 1

	Group	EV1	EV2	EV3	EV4
Input 1	1	-1	-1	1	1
Input 2	1	-1	-1	1	1
Input 3	1	-1	1	-1	1
Input 4	1	-1	1	-1	1
Input 5	1	1	-1	-1	1
Input 6	1	1	-1	-1	1
Input 7	1	1	1	1	1
Input 8	1	1	1	1	1



Contrasts 3 F-tests 3

		Title	EV1	EV2	EV3	EV4	F1	F2	F3
C1	☑	A	1	0	0	0	☑	☐	☐
C2	☑	B	0	1	0	0	☐	☑	☐
C3	☑	AB	0	0	1	0	☐	☐	☑



C1	A	1	0	0	0	■	□	□
C2	B	0	1	0	0	□	■	□
C3	AB	0	0	1	0	□	□	■

Random Effects

If both factors are random effects then the F-tests for the effects of the factors are different - the denominator in the F is derived from the interaction effect and not the within-cell errors. In this case, the relevant F images for factors A and B can be formed as Fa=fstat1/fstat3 and Fb=fstat2/fstat3. In order to carry this out, first run FEAT using the above design. Then:

```
cd <featdir>/stats

avwmaths fstat1 -div fstat3 fstata
avwmaths fstat2 -div fstat3 fstatb

ftoz -zout zfstata fstata 1 1
ftoz -zout zfstatb fstatb 1 1
```

You could then do thresholding on zfstata and zfstatb with **easythresh**.

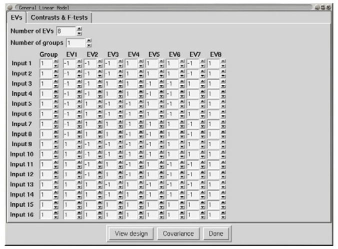
Mixed Effects

If one factor is random and the other is fixed then we want a mixed effects analysis. In this case the fstat which needs the different denominator is the one associated with the **fixed** factor. For example, if factor A is fixed and factor B is random, then fstat2 already gives you the effect of factor B and for factor A you need to create Fa=fstat1/fstat3 as above.

ANOVA : 3-factors 2-levels

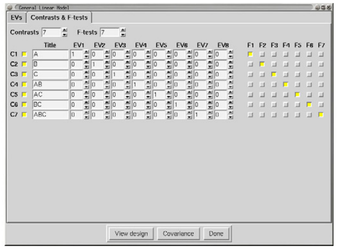
Fixed Effects

We have 16 subjects and 3 factors, each at 2 levels. To carry out a standard ANOVA we use the following, the F-tests giving the standard ANOVA results for the factors and their interactions.



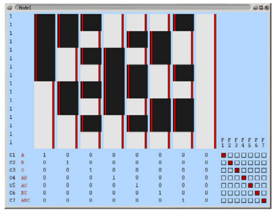
Number of EVs 8
Number of groups 1

	Group	EV1	EV2	EV3	EV4	EV5	EV6	EV7	EV8
Input 1	1	-1	-1	-1	1	1	1	-1	1
Input 2	1	-1	-1	-1	1	1	1	-1	1
Input 3	1	-1	-1	1	1	-1	-1	1	1
Input 4	1	-1	-1	1	1	-1	-1	1	1
Input 5	1	-1	1	-1	-1	1	-1	1	1
Input 6	1	-1	1	-1	-1	1	-1	1	1
Input 7	1	-1	1	1	-1	-1	1	-1	1
Input 8	1	-1	1	1	-1	-1	1	-1	1
Input 9	1	1	-1	-1	-1	-1	1	1	1
Input 10	1	1	-1	-1	-1	-1	1	1	1
Input 11	1	1	-1	1	-1	1	-1	-1	1
Input 12	1	1	-1	1	-1	1	-1	-1	1
Input 13	1	1	1	-1	1	-1	-1	-1	1
Input 14	1	1	1	-1	1	-1	-1	-1	1
Input 15	1	1	1	1	1	1	1	1	1
Input 16	1	1	1	1	1	1	1	1	1



Contrasts 7 F-tests 7

		Title	EV1	EV2	EV3	EV4	EV5	EV6	EV7	EV8	F1	F2	F3	F4	F5	F6	F7
C1	☑	A	1	0	0	0	0	0	0	0	☑	☐	☐	☐	☐	☐	☐
C2	☑	B	0	1	0	0	0	0	0	0	☐	☑	☐	☐	☐	☐	☐
C3	☑	C	0	0	1	0	0	0	0	0	☐	☐	☑	☐	☐	☐	☐
C4	☑	AB	0	0	0	1	0	0	0	0	☐	☐	☐	☑	☐	☐	☐
C5	☑	AC	0	0	0	0	1	0	0	0	☐	☐	☐	☐	☑	☐	☐
C6	☑	BC	0	0	0	0	0	1	0	0	☐	☐	☐	☐	☐	☑	☐
C7	☑	ABC	0	0	0	0	0	0	1	0	☐	☐	☐	☐	☐	☐	☑



C1	A	1	0	0	0	0	0	0	0	■	□	□	□	□	□	□
C2	B	0	1	0	0	0	0	0	0	□	■	□	□	□	□	□
C3	C	0	0	1	0	0	0	0	0	□	□	■	□	□	□	□
C4	AB	0	0	0	1	0	0	0	0	□	□	□	■	□	□	□
C5	AC	0	0	0	0	1	0	0	0	□	□	□	□	■	□	□
C6	BC	0	0	0	0	0	1	0	0	□	□	□	□	□	■	□
C7	ABC	0	0	0	0	0	0	1	0	□	□	□	□	□	□	■

Random/Mixed Effects

The following table shows how to test for factor effects with various models:

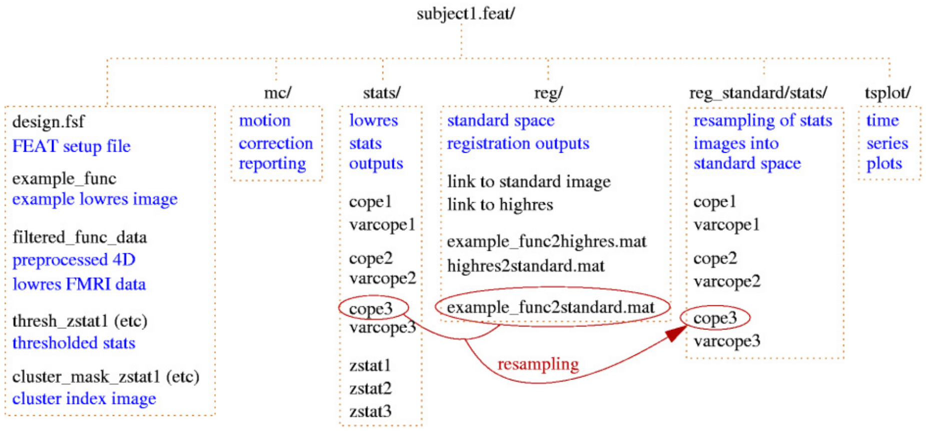
model	A	B	C	F(A)	F(B)	F(C)	F(AB)	F(AC)	F(BC)	F(ABC)
1	F	F	F	fstat1	fstat2	fstat3	fstat4	fstat5	fstat6	fstat7
2	R	R	R				fstat4/fstat7	fstat5/fstat7	fstat6/fstat7	fstat7
3	F	R	R		fstat2/fstat6	fstat3/fstat6	fstat4/fstat7	fstat5/fstat7	fstat6	fstat7
4	F	F	R	fstat1/fstat5	fstat2/fstat6	fstat3	fstat4/fstat7	fstat5	fstat6	fstat7

FEAT Output

FEAT finds the directory name and filename associated with the 4D input data. If the associated directory is writable by you then a related whatever.feat directory is created into which all FEAT output is saved. If not, the FEAT directory is created in your home directory. In either case, if the appropriately named FEAT directory already exists, a "+" is added before the .feat suffix to give a new FEAT directory name. (Of course, all the above gets ignored if you explicitly set the output directory name.)

If you rerun **Post-stats** or **Registration**, you can choose (under the **Misc** tab) whether to overwrite the relevant files in the chosen FEAT directory or whether to make a complete copy of the FEAT directory and write out new results in there.

All results get saved in the FEAT directory.



- **cluster_mask_zstat1.hdr** image of clusters found for contrast 1; the values in the clusters are the index numbers as used in the cluster list.
- **cluster_zstat1.html / .txt** the list of significant clusters for contrast 1. Note that the co-ordinates are the original voxel co-ordinates. (MEDx users: MEDx inverts y, so to use the y values in MEDx, use y_medx = y_size - 1 - y_feat where y_size is the size of the image in the y direction.)
- **cluster_zstat1_tal.html / .txt** the same, but with co-ordinates given in Talairach space. This exists if registration to Talairach space has been carried out.
- **design.con** list of contrasts requested.
- **design.fsf** FEAT setup file, describing everything about the FEAT setup. This can be loaded into the FEAT GUI.
- **design.fts** list of F-tests requested.
- **design.gif** 2D image of the design matrix.
- **design.mat** the actual values in the design matrix.
- **design.trg** event onset times, created to be used in peri-stimulus timing plots.
- **design_cov.gif** 2D image of the covariance matrix of the design matrix.
- **example_func.hdr** the example functional image used for colour rendering, and also the one that was used as the target in motion correction. This is saved before any processing is carried out. This is also the image that is used in registration to structural and/or standard images.
- **example_func2highres.*** files are related to the registration of the low res FMRI data to the high res image. The .xfm and .mat files are the transformation files in MEDx and raw format respectively. The .gif image includes several slices showing overlays of the two images combined after registration. (Note that the inverse of each transform file is also saved (e.g. highres2example_func.mat) to make it easy for you later to take standard or highres images back into the lowres space.)
- **example_func2standard.*** files are related to the registration of the low res FMRI data to the standard image.
- **filtered_func_data.hdr** the 4D FMRI data after all filtering has been carried out. (prefiltered_func_data.hdr is the output from motion correction and the input to filtering, and will not normally be saved in the FEAT directory.) Although filtered_func_data.hdr will normally have been temporally high-pass filtered, it is not zero mean; the mean value for each voxel's timecourse has been added back in for various practical reasons. When FILM begins the linear modelling, it starts by removing this mean.
- **highres.hdr** is a symbolic link to the high res image.
- **highres2standard.*** files are related to the registration of the high res image to the standard image.
- **lmax_zstat1.txt** and **_tal.txt** are lists of local maxima within clusters found when thresholding.
- **mask.hdr** the binary brain mask used at various stages in the analysis.
- **mc/prefiltered_func_data_mcf.par** a text file containing the rotation and translation motion parameters estimated by MCFLIRT, with one rwo per volume.

- **mc/mc_rot.gif**, **mc/mc_trans.gif** plots showing these parameters as a function of volume number (i.e., time).
- **rendered_thresh_zstat1.png** 2D colour rendered stats overlay picture for contrast 1.
- **rendered_thresh_zstat1.hdr** 3D colour rendered stats overlay image for contrast 1. After reloading this image, use the **Statistics Colour Rendering** GUI to reload the colour look-up-table.
- **report.log** a log of all the programs that the feat script ran (ie the same as **report.log** but without the log outputs).
- **report.html** the web page FEAT report (see below).
- **report.log** a log of the FEAT run, including all calls to FSL programs and their log outputs.
- **standard.hdr** is a symbolic link to the standard image.
- **stats/contrastlogfile** a logfile showing how well the statistics fit a Gaussian distribution, on the assumption of no activation.
- **stats/cope1.hdr** the contrast of parameter estimates image for contrast 1.
- **stats/corrections** a list of statistical corrections used within FILM modelling.
- **stats/dof** the mean estimated degrees-of-freedom over the whole data set.
- **stats/neff1.hdr** a statistical correction image for contrast 1.
- **stats/glslogfile** a FILM run logfile.
- **stats/pe1.hdr** the parameter estimate image for EV 1.
- **stats/probs** a list of probabilities used for estimating Gaussian statistical fitting.
- **stats/ratios** a list of estimates for Gaussian statistical fitting.
- **stats/sigmasquareds.hdr** the 3D image summarising the residuals (errors) in the linear model fitting.
- **stats/smoothness** the estimation of the smoothness of the 4D residuals field, used in inference.
- **stats/stats.txt** another FILM logfile.
- **stats/threshac1.hdr** The FILM autocorrelation parameters.
- **stats/tstat1.hdr** the T statistic image for contrast 1 (=cope/sqrt(varcope)).
- **stats/varcope1.hdr** the variance (error) image for contrast 1.
- **stats/zstat1.hdr** the Z statistic image for contrast 1
- **thresh_zstat1.hdr** the thresholded Z statistic image for contrast 1.
- **tsplot/tsplot1.gif** 2D picture of the model vs data plot for the maximum Z statistic voxel from contrast 1.
- **tsplot/tsplot1.txt** text file of values of the model vs data plot for the maximum Z statistic voxel from contrast 1. The first column is the data, the second column is the partial model fit for contrast 1, the third column is the full model fit and the fourth column is the reduced data for contrast 1 (see below).
- **tsplot/tsplotc1.gif** 2D picture of the model vs data plot averaged over all voxels in all significant clusters for contrast 1.
- **tsplot/tsplotc1.gif** text file of values of the model vs data plot averaged over all voxels in all significant clusters for contrast 1.
- **tsplot/tsplot1p.gif / tsplot/tsplotc1p.gif** plots of reduced data vs cope partial model fit - i.e. data-full_model+partial_model vs partial_model.

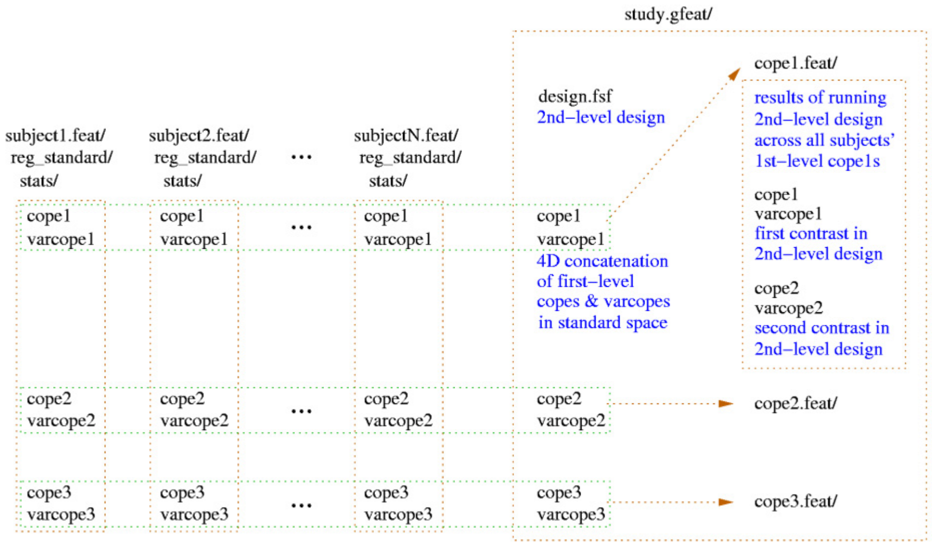
If you have run F-tests as well as T-tests then there will also be many other files produced by FEAT, with filenames similar to those above, but with **zfstat** appearing in the filename.

The web page report includes the motion correction plots, the 2D colour rendered stats overlay picture for each contrast, the data vs model plots, registration overlay results and a description of the analysis carried out.

Higher-Level FEAT Output

A second-level .gfeat directory contains one 4D cope*.hdr for each of the first-level contrasts; these are simply the concatenation (across the first-level analyses) of those first-level cope images (in standard space) - ie the 4th dimension here is the number of first-level analyses. This is the input to the second-level analysis.

After the second-level analysis has completed each of those 4D cope*.hdr files in the .gfeat directory will have resulted in a .feat second-level output, containing all analysis steps (and of course, second-level output copes).



FEAT Programs

- **Feat** - the FEAT GUI, as described above.
- **feat** - the FEAT script which the GUI calls when the setup has been completed and **GO** is pressed. If you have saved a FEAT setup file (eg design.fsf) to file, then you can run FEAT without the GUI by typing **feat <featsetupfile.fsf>**.
- **easythresh** - a simple script for carrying out cluster-based thresholding and colour activation overlaying.
- **featregapply** - a program which applies the lowres to standard-space registration transforms to the lowres stats images in a FEAT directory to generate standard-space versions, for inputting to higher-level analysis.
- **Featquery** (GUI) and **featquery** (command-line script) - a program which allows you to interrogate FEAT results by defining a mask or set of co-ordinates (in standard-space, highres-space or loweres-space) and get mean stats values and time-series.
- **Featwatcher** (GUI) - a simple GUI that presents the progress of a FEAT analysis.
- **feat_model** - a program which uses the FEAT setup file to create the required design matrix as a text file, as well as the related design matrix pictures.
- **mccutup** - convert the estimates of motion saved in a FEAT directory to separate files which can then be fed into FEAT as **Custom (1 entry per volume)** EVs.
- **Renderhighres** (GUI) and **renderhighres** (command-line script) - transforms all thresholded stats images in a FEAT directory into high resolution or standard space and overlays these onto the high resolution or standard space images. This then produces PNG format pictures of the overlays and, by default, deletes the 3D AVW colour overlay images.
- **tsplot** - create time series plots for a FEAT directory.

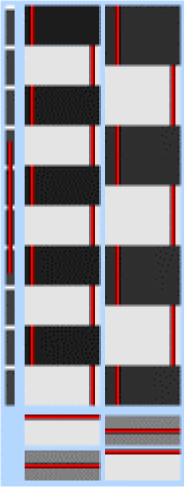
Appendix A: Brief Overview of GLM Analysis

General Linear Modelling (more correctly known simply as "linear modelling") sets up a model (i.e., what you expect to see in the data) and fits it to the data. If the model is derived from the stimulation that was applied to the subject in the MRI scanner, then a good fit between the model and the data means that the data was indeed caused by the stimulation.

The GLM used here is univariate. This means that the model is fit to each voxel's time-course separately. (Multivariate would mean that a much more complex analysis would take place on all voxels' time-courses at the same time, and interactions between voxels would be taken into account. Independent Components Analysis - ICA - is an example of multivariate analysis.) For the rest of this section, you can imagine that we are only talking about one voxel, and the fitting of the model to that voxel's timecourse. Thus the data comprises a single 1D vector of intensity values.

A very simple example of linear modelling is **y(t)=a*x(t)+b+e(t)**. **y(t)** is the data, and is a 1D vector of intensity values - one for each time point, i.e., is a function of time. **x(t)** is the model, and is also a 1D vector with one value for each time point. In the case of a square-wave block design, **x(t)** might be a series of 1s and 0s - for example, 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 etc. **a** is the parameter estimate for **x(t)**, i.e., the value that the square wave (of height 1) must be multiplied by to fit the square wave component in the data. **b** is a constant, and in this example, would correspond to the baseline (rest) intensity value in the data. **e** is the error in the model fitting.

If there are two types of stimulus, the model would be **y=a1*x1+a2*x2+b+e**. Thus there are now two different model waveforms, corresponding to the two stimulus timecourses. There are also two interesting parameters to estimate, **a1** and **a2**. Thus if a particular voxel reponds strongly to model **x1** the model-fitting will find a large value for **a1**; if the data instead looks more like the second model timecourse, **x2**, then the model-fitting will give **a2** a large value.



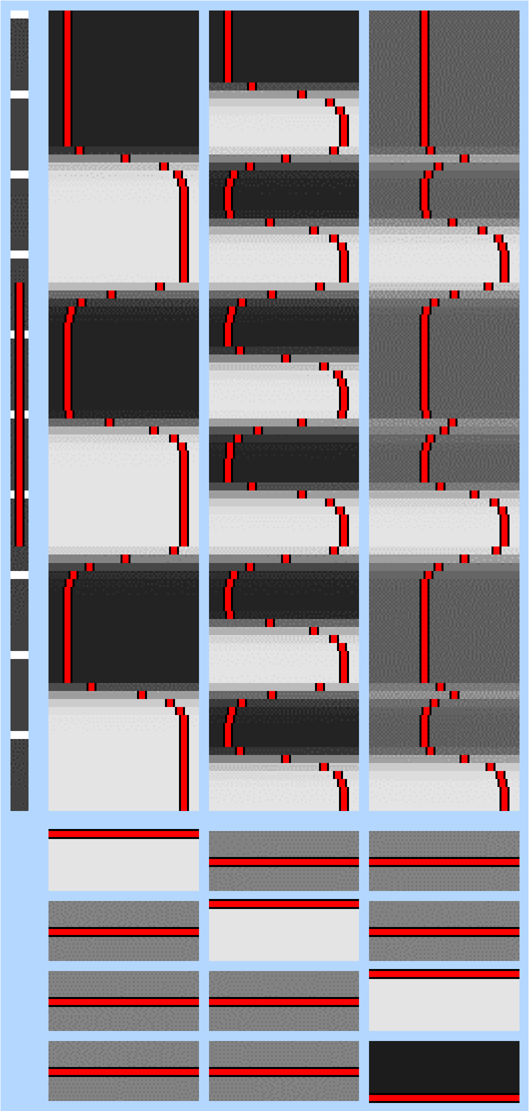
GLM is normally formulated in matrix notation. Thus all of the parameters are grouped together into a vector **A**, and all of the model timecourses are grouped together into a matrix **X**. However, it isn't too important to follow the matrix notation, except to understand the layout of the design matrix, which is the matrix **X**. The main part of the image shown here contains two such model timecourses. Each column is a different model timecourse, with time going down the image vertically. Thus the left column is **x1**, for example, the timecourse associated with visual stimulation, and the right column is **x2**, e.g., auditory stimulation, which has a different timecourse to the visual stimulation. Note that each column has two representations of the model's value - the black->white intensity shows the value, as does the red line plot. Make sure that you are comfortable with both representations.

When the model is fit to the data, for each voxel there will be found an estimate of the "goodness of fit" of each column in the model, to that voxel's timecourse. In the visual cortex, the first column will generate a high first parameter estimate, and the second column will generate a low second parameter estimate, as this part of the model will not fit the voxel's timecourse well. Each column will be known as an explanatory variable (EV), and in general represents a different stimulus type.

To convert estimates of parameter estimates (PEs) into statistical maps, it is necessary to divide the actual PE value by the error in the estimate of this PE value. This results in a **t** value. If the PE is low relative to its estimated error, the fit is not significant. Thus **t** is a good measure of whether we can believe the estimate of the PE value. All of this is done separately for each voxel. To convert a **t** value into a **P** (probability) or **Z** statistic requires standard statistical transformations; however, **t**, **P** and **Z** all contain the same information - they tell you how significantly the data is related to a particular EV (part of the model). **Z** is a "Gaussianised **t**", which means that a Z statistic of 2 is 2 standard deviations away from zero.

As well as producing images of **Z** values which tell you how strongly each voxel is related to each EV (one image per EV), you can compare parameter estimates to see if one EV is more "relevant" to the data than another. This is known as contrasting EVs, or producing contrasts. To do this, one PE is subtracted from another, a combined standard error is calculated, and a new Z image is created. All of the above is controlled by you, by setting up contrasts. Each output Z statistic image is generated by setting up a contrast vector; thus set the number of outputs that you want. To convert a single EV into a Z statistic image, set it's contrast value to 1 and all others to 0. Thus the simplest design, with one EV only, has just one contrast vector, and only one entry in this contrast vector: 1. To compare two EVs, for example, to subtract one stimulus type (EV 1) from another type (EV 2), set EV 1's contrast value to -1 and EV 2's to 1. A Z statistic image will be generated according to this request, answering the question "where is the response to stimulus 2 significantly greater than the response to stimulus 1?"

The bottom part of the above image shows the requested contrasts; each column refers to the weighting of the relevant EV (often either just 1, 0 or -1), and each row is a different contrast vector. Thus each row will result in it's own Z statistic image. Here the contrasts are [1 0] and [0 1]. Thus the first Z stat image produced will show response to stimulus type 1, relative to rest, and the second will show the response to stimulus type 2.



If you want to model nonlinear interactions between two EVs (for example, when you expect the response to two different stimuli when applied simultaneously to give a greater response than predicted by adding up the responses to the stimuli when applied separately), then an extra EV is necessary. The simplest way of doing this is to setup the two originals EVs, and then add an interaction term, which will only be "up" when both of the original EVs are "up", and "down" otherwise. In the example image, EV 1 could represent the application of drug, and EV 2 could represent visual stimulation. EV 3 will model the extent to which drug+visual is greater than the sum of drug-only and visual-only. The third contrast will show this measure, whilst the fourth contrast [0 0 -1] shows where negative interaction is occurring.

All of the EVs have to be independent of each other. This means that no EV can be a sum (or weighted sum) of other EVs in the design. The reason for this is that the maths which are used to fit the model to the data does not work properly unless the design matrix is of "full rank", i.e. all EVs are independent. A common mistake is to model both rest and activation waveforms, making one an upside-down version of the other; in this case EV 2 is -1 times EV 1, and therefore linearly dependent on it. It is only necessary to model the activation waveform.

With "parametric designs", there might be several different levels of stimulation, and you probably want to find the response to each level separately. Thus you should use a separate EV for each stimulation level. (If, however, you are very confident that you know the form of the response, and are not interested in confirming this, then you can create a custom waveform which will match the different stimulation levels, and only use one EV.) If you want to create different contrasts to ask different questions about these responses, then: [1 0 0] shows the response of level 1 versus rest (likewise [0 1 0] for level 2 vs rest and [0 0 1] for level 3). [-1 1 0] shows where the response to level 2 is greater than that for level 1. [-1 0 1] shows the general linear increase across all three levels. [1 -2 1] shows where the increase across all three levels deviates from being linear (this is derived from (l3-l2)-(l2-l1)=l3-2*l2+l1).

Thus there often exists a natural hierarchy in contrast vectors. In the above example, [1 1 1] shows overall activation, [-1 0 1] shows linear increase in activation and [1 -2 1] shows (quadratic) deviation from the linear trend. Note that each contrast is orthogonal to the others (e.g. -1*1 + 0*1 + 1*1 = 0) - this is important as it means that each is independent of the others. A common mistake might be, for example, to model the linear trend with [1 2 3], which is wrong as it mixes the average activation with the linear increase.

Appendix B: Design Matrix Rules

This section describes the rules which are followed in order to take the FEAT setup and produce a design matrix, for use in the FILM GLM processing.

Here HTR model means "high temporal resolution" model - a time series of values that is used temporarily to create a model and apply the relevant HRF convolution before resampling down in time to match the temporal sampling of the FMRI data.

Note that it is assumed that every voxel was captured instantaneously in time, and at the same time, exactly halfway through a volume's time period, not at the beginning. This minimises timing errors, if slice-timing correction has not been applied.

No constant column is added to the model - instead, each EV is demeaned, and each voxel's time-course is demeaned before the GLM is applied.

```
for each EV
(
  if ( square waveform )
    fill HTR model with 0s or 1s
  else if ( sinusoidal waveform )
    fill HTR model with sinusoid scaled to lie in the range 0:1
  else if ( custom waveform )
    fill HTR model with custom information, with 0s outside of
    specified periods

  demean HTR

  create "triggers" i.e. record the start and end of event or block

  create blurring+delaying HTR HRF convolution kernel, normalised so
  that the sum of values is 1 (in the case of basis functions, several
  related kernels are created)

  convolve HTR model with HRF convolution kernel (values in HTR model
  for t<0 are set to 0 to allow simple convolution)
    OR
  in the case of sinusoidal original waveform; create harmonics (if
  requested)

  subsample HTR model to match the temporal resolution of the data;
  take the value in the centre of each volume's time period

  apply the same high-pass temporal filtering that was applied to the
  data

  re-demean
  orthogonalise current EV wrt earlier EVs if requested (form
  temporary matrix from selected EVs, carry out SVD, and subtract
  projection of current EV onto each vector in SVD output)

  instead of all the above - if this EV is an "interaction" (nonlinear
  interaction between other EVs);
    model = PRODUCT(other EVs, after subtracting the min value from each)

  if requested, create a new EV, calculated as the temporal derivative
  of the current EV

  re-demean (again!)
)
```

Featquery - FEAT Results Interrogation User Guide

Introduction

Featquery is a script which allows you to interrogate FEAT results by defining a mask (or co-ordinates of a single-voxel) in standard-space, highres-space or lowres-space, to get mean, max etc stats values and time-series within that mask or at that voxel position. For example, you might define a standard-space mask for the motor cortex, and Featquery will tell you the mean (and peak) % signal change associated with your modelled experimental paradigm within that area.

First you must select a previously-created FEAT output directory whose results you wish to investigate. You can run multiple queries by changing the **Number of FEAT directories**. The results of running Featquery will be saved in a directory ("featquery") inside each FEAT directory chosen. If you have already run Featquery on a FEAT directory, a "+" will be appended to the subdirectory name, e.g. "featquery+".

Choosing stats images to investigate

Once you have selected a FEAT directory, a list of all possibly interesting stats images inside that FEAT directory will appear on the Featquery GUI. Turn on the ones you are interested in. (If you turn on **filtered_func_data** you will be given mean (and max etc) stats values within the defined mask, including searching over time as well as space.)

If you select **Convert PE/COPE values to %**, any PE or COPE parameter estimate or contrast values will be converted to percentage change values before reporting. This is achieved by dividing the PE/COPE values by the mean image from filtered_func_data. **Warning**: this % is based on the **assumption** that the "height" of the model waveform is 1, which in general it is for FEAT-created block designs, but in general is **not** for event-related designs or custom waveforms. In order to get a true % change value you must multiply the output by the height of the relevant model waveform (see the design.mat file). In the case of contrasts (COPEs), the interpretation of this % needs even more careful thought.

Setting up a mask or voxel co-ordinates

You must now choose a **Mask image**. This would normally be a binary image in standard-space, highres-space or lowres-space, with a region-of-interest (ROI), for example, the visual cortex, created by any method. Featquery will automatically detect which space this mask is in (ie standard-space, highres-space or lowres-space) and will transform it into the native lowres space of example_func; of course this can only work if FEAT registration was setup and carried out.

If you want a different mask for each selected FEAT directory, specify a mask name as a relative filename (ie without a "/"). This mask will then be looked for relative to each FEAT directory.

Alternatively, you can specify a single position (in voxels or mm) at which to extract values from the chosen stats images. This still requires a "mask" image to be chosen, as it is relative to this mask image that the co-ordinates have meaning. Thus if you want the co-ordinates to be in lowres space, just select "mask" or "example_func"; if you want to specify standard-space co-ordinates ("Talairach space") then choose "reg/standard". It is wise to be careful here - after running Featquery, have a look at the created featquery/mask image to check that the voxel finally chosen is in a sensible place!

Advanced options

If you want to only allow stats values above a threshold to enter into the calculations (of mean, max etc) then turn on **Apply threshold** and select a value.

If you have selected a mask image in standard or highres space, this will get transformed into lowres space as described above. This involves interpolation; at the edges of the mask there will be a continuous range of values from 1 down to 0. In order to get back to a binary mask, this must be thresholded at some value - the default is 0.5. However, if you want the mask to be slightly more or less inclusive than that default, you can **Change mask post-interpolation thresholding** - for example, by reducing the value to 0.3, the final lowres mask will be slightly larger.

Go

When you press **Go**, Featquery produces all the requested stats values including mean, min, max and position of max. These get logged to the file report.txt inside the featquery subdirectory inside the FEAT directory, and also copied to the screen. Also, various timeseries textfiles are saved; one for the mean timeseries over the mask, and also one for each of the positions of the max value of each of the stats images.

Featwatcher - FEAT Progress Monitor User Guide

Featwatcher is a simple GUI that presents the progress of a FEAT analysis. When you start FEAT running with the **Go** button, it automatically brings up the Featwatcher GUI. At the top is the name of the FEAT output directory; under that is a small panel which shows some running statistics on the currently running FEAT sub-process - %CPU usage, total CPU time so far and memory usage. In the large panel is shown the **report.log** file which lists all commands run by FEAT, along with their output.

You can also run **Featwatcher <feat_directory_name.feat>** from the command line, to view the progress in an existing running FEAT directory.

Note that FEAT still keeps running if you exit Featwatcher or the FEAT GUI, and even, in general, if you completely log out of your computer.