# Using FSL to process FMRI data – week 4

## Introduction

This workshop has been adapted with permission from the freely available introductory workshop on the Oxford FMRIB FSL website. I have removed a few things to save time and added slightly more information in some places. I’ve also reorganised the material from the website so that essential things are in the first part of this document and more advanced and technical things are in the second (optional) part.

## To start..

A few things worth keeping in mind:

* <https://eroesch.github.io/2019-20-PYM0FM/> is the course website, which I will use more than BlackBoard
* <http://www.reading.ac.uk/internal/its/help/its-help-networks/pcsecurity-setupvpn.aspx> gives all the information you need to access the University VPN (<https://simple.wikipedia.org/wiki/Virtual_private_network>). You will need to be on the VPN to access the CINN Nutanix VM platform from your own computer.
* <https://rrc.reading.ac.uk:9440> is the portal to the CINN Nutanix Virtual Machines, which you can use to deploy your own computer.

Remember: When in doubt. Google it.

Assuming you have completed the tutorial on bash shell, and have successfully launched a CINN Virtual Machine, you can find the folder we will use for PYM0FM using the command “cd” and list the files, and folders within that folder using the command “ls” (that’s a lower-case L and a lower-case s).

Not also that in the grey box, the “$” represents the prompt for the bash, i.e. where you will type the command; you should **not** type the “$”, just type what comes afterwards, as it is shown.

$ cd /storage/silver/pym0fm/

$ ls

At this point, bash should show you a list of folders, including one that has named after your IT login.

## Find the data

In the command terminal, if you haven’t done it already, go to your folder on the PYM0FM folder, by typing the below, and replacing your ITS login accordingly

$ cd /storage/silver/pym0fm/<YOUR ITS LOGIN>

Then use the “ls” (that’s an “l” as in lullaby) command to see a list of files in the directory, simply type:

$ ls

You will see.... nothing. ☺ This is your folder, where you will store your data and analyses. At the moment, there is nothing to show. Let’s start by adding some data, type:

$ cp ../course\_material/fsl\_course\_data.tar.gz .

$ ls

You will now see that, inside your folder, there is one file. The “cp” command copied a file from a different folder, using a path relative to where you are currently standing: Assuming you are in your folder “..” indicates to bash that it should go one folder up, and then go to ../course\_material. The final “.” at the end of the command tells “cp” that it should copy the file where you are currently standing.

$ ls -al

This adds parameters to the “ls” command, to show details about the files (using “-l”) and also to show any hidden files (using “-a”): you can, but you don’t have to, combine the two parameters “-l” and “-a” into a single set of parameter, into “-la”. This gives you more information about the file we have just copied inside your folder. It should show something like the following:

-rw-r--r-- 1 tz901918 stor-silver-pym0fm 5348460464 Oct 22 13:16 fsl\_course\_data.tar.gz

The output above is about the file named “fsl\_course\_data.tar.gz”, which is contained in the original folder /storage/silver/pym0fm/course\_material/. I created it, and my IT login is tz901918.

When you type “ls -al” inside your own folder, provided that you typed the “cp” command above as instructed, you should see a copy of the file fsl\_course\_data.tar.gz that has appeared inside your folder.

The “ls -al” command will give you quite a lot more information than just the name. You can see that the file is about 5GB large, 5,348,460,464 bytes to be precise. You can see that you have created it (as part of the group stor-silve-pym0fm), and are thus the owner of the file, and you can see that it has the following permissions: -rw-r--r-- which should read in groups of 3 letters:

* The first dash – says it is not a directory, and it’s a file, otherwise it would read “d” (for directory)
* The second group of 3 letters corresponds to the permissions you personally have over this file: “rw-“ means you can read and write, but cannot execute the file.
* The second group of 3 letters corresponds to the permissions any users belonging to your group (stor-silver-pym0fm) have over my file: “r--” means they (anybody else in the calss) have read permission, but they can’t write or execute the file.
* The third group of 3 letters corresponds to the permissions anybody else (who is not me, and not in the same group as you, i.e. anybody else in CINN) have over your file: “r--” which means they have read permissions, but cannot write or execute the file.

This file is compressed. That means it is an entire folder compressed into a single file that will take much less space than if it was not compressed. Yes, the file is currently 5GB, and the uncompressed folder will be larger. In your folder, type:

$ tar xvzf fsl\_course\_data.tar.gz

After a little while, the command will have uncompressed the file, and created a folder called “fsl\_course\_data”, which contains a selection of data for you to play with. You can always repeat this procedure if you want to start anew.

Exercise: Visit the folder, and try to figure out what this is.

## FSLView

On the command line, type:

$ module avail

“module” is a little tool that provides the ability to configure the system. In CINN, we use it to make sure that we can access several versions of the same software, without them colliding with each other. The command “module avail” shows all the modules that are available. Type:

$ module load fsl5.0.9

This will load the module for fsl version 5.0.9. There is also a second version available, v6, but for PYM0FM, we will stick to using v5.

Start FSLView by typing

$ fslview &

The & means that the program you asked for (fslview) runs in the background in the terminal (or shell), and you can keep typing other commands while it is running. If you had not done that then you would not be able to do anything else in the terminal until you killed fslview (or, alternatively, you could type control-z in the terminal and then bg to get fslview running in the background post-hoc).

Load in the image example\_func.nii.gz, by pressing File -> Open and selecting the image.

Hold the mouse button down in one of the view panels and move it around - see how various things update as you do so:

* the other panels update their view
* the cursor's position in both voxel and mm co-ordinates gets updated (hover your mouse over the X Y and Z coordinate boxes to find out from “tooltips” which boxes give the values in voxels, and which boxes give the values in mm)
* the image intensity at the cursor is shown

Mode Toolbar:     mode

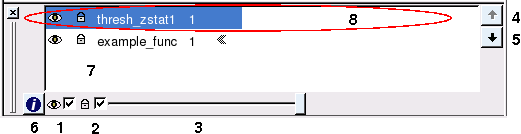
Now click several times on the small upwards arrow [1] (on some computers the arrows are replaced with a small "+" and "-") by the scale control (currently showing "100%"). This is one way of controlling the zoom. If you click on the icon of a hand [3] you can drag a view around. You can reset the view settings by pressing on the button with the arrow on top of the magnifying glass [6]. You can also set the field-of-view of a panel by clicking on the button with the magnifying glass [5], and then click-and-dragging the mouse in a view panel. Return to "normal view mode" (as opposed to "move" or "zoom" mode) by clicking on the button with the small cross [2].

Bricon Toolbar:     

You can control the brightness and contrast either by using the wheels/sliders [3] near the "brightness/contrast" button [2] (this button allows you to reset the brightness and contrast to their initial settings). You can also control the intensity display range explicitly by typing numbers into the Min and Max boxes [1].

Open a lightbox view using Tools -> Lightbox. This will produce a set of axial (transverse) slices. If you drag the mouse around in the viewer you can see that the cursor position is linked in the two views of the data. This is particularly useful when you have several images loaded in at the same time.

If you increase the % zoom in the lightbox mode sufficiently, you will get a scrollbar to let you control which slices you are viewing. Kill the lightbox view now and expand the ortho view back to full size.

Layer Toolbar:     

Now load in a second image using File -> Add; thresh\_zstat1 is a thresholded FMRI stats image. In the bottom-right panel is a list of loaded images - the layer toolbar. Images can be turned on and off by double-clicking in that list [7] (or by highlighting the image in the list [8] and then clicking on the checkbox by the small eye icon [1] at the bottom). Once highlighted in the image list, an image's transparency can be changed with the slider [3] (at the bottom of the box); reduce the "solidity" of the stats overlay image.

Also, when an image in the image list is highlighted:

* You can click on the i button [6] and change the image's colour display scheme (LUT - look-up-table), and see/change some other basic image information.
* You can set the brightness/contrast settings for that image, as described above.
* You can view voxel intensities for that image in the normal place (to the right of the voxel co-ordinate displays).

View Toolbar:     view-tool

Now close down all views; all loaded images have now been deleted from FSLView. Next, load in filtered\_func\_data, a 4D FMRI time series. Now watch this as a movie by pressing the film-strip button [1] that appears when a 4D file is loaded as the primary image (the first image to be loaded). You will see the time-counter ("Volume") scrolling through the different time-point values. Note that whilst the movie is running you can still change the cursor position. You can even re-load the thresh\_zstat1 stats overlay image back in whilst the movie is running. Stop the movie by pressing the movie button again.

Looking at a movie of your 4D time series data is a very good way of spotting problems with the data such as excessive head motion, sudden image intensity changes (spikes), and ghosting. It is always worth doing this as an initial quality check on your data.

To illustrate this, load in an a 4D FMRI time series acquired while one of my collaborators dozed off in the scanner instead of attending to the experiment. Close down the current view and load the file headMotion and watch it as a movie.

These images have not had any motion correction or smoothing applied, they are “fresh from the scanner”, unlike the last movie you viewed where the images had already been preprocessed.

Timeseries Toolbar:     timeseries-tool

Close the headMotion data series and open filtered\_func\_data again. Open a timeseries graph view with Tools -> Timeseries (you will need to have the 4D dataset highlighted in the image list for this option to be active). Move around with the mouse button held down until you are on a "highly activated" voxel. Press the + button [1] in the timeseries view and that voxel's timeseries gets saved in the display. You can add more timeseries by moving to new voxels and pressing + again.

If you have used "+" to add several timeseries to the display they may all have quite different mean values, making comparison difficult - press the "-μ" button [3] to demean all viewed timeseries. Once you have turned on demeaning you can also choose to turn on the "%" button [4], which shows the y-axis as a % of the mean level (i.e. in this case you're now seeing the BOLD % signal change values).

Histogram Toolbar:     histogram-tool

To view a basic histogram of the currently displayed image press Tools -> Image histogram. A histogram plots the distribution of intensities in the image; image intensity is on the x axis and the number of voxels with that intensity is on the y axis. View the histogram of example\_func; you can click in the histogram to view exact values, which appear at the bottom left. Click on the toolbox button [3] to get more histogram options. If you have time take a look at some voxel time series plots and an image histogram for the headMotion file.

Finally, we'll look at some more advanced FMRI analysis features in FSLView. Kill FSLView and enter the following (remember to use the tab key to speed up typing)

$ cd /storage/silver/pym0fm/<YOUR ITS LOGIN>/fsl\_course\_data/fmri/ptt/at/at\_left.feat

You are now in an example FMRI analysis FEAT output directory. Note, FEAT is the program that you will use next week to find activations in 4D functional data. Type

$ fslview filtered\_func\_data thresh\_zstat1 thresh\_zstat2 &

The image names on the fslview command line tell FSLView to load in each of these images, to save you having to load them in by hand in the GUI. Because FSLView detects that you are in a FEAT directory it automatically turns on timeseries viewing. You might want to resize the FSLView window (make it wider) and then auto-resize the inner windows with Window -> Tile.

One of the nice features when in "FEAT mode" is that you can see a plot of the fitted timeseries model versus the data - in the timeseries window change No model to Full model only (timeseries toolbar [6]). Now when you click around in the image you can see not just the data timeseries but also the fitted model.

The other nice feature is the cluster browser - turn this on with Tools -> Cluster browser. This lets you select a stats image name and get summary information about the different activation clusters found by FEAT. You can even click on a row in the cluster table and FSLView will take you to the peak voxel in that cluster

This FEAT output directory already contains versions of some of the images upsampled to "standard space" - the common image space that multisubject analyses are carried out in. Kill fslview and run: (note: only press return once you have typed in both lines of this)

$ fslview reg\_standard/reg/highres.nii.gz reg\_standard/stats/cope1 -l Red-Yellow -b 100,400

The options tell FSLView what colourmap to use for the FMRI activation effect size image cope1, and what intensity display range. Note that the Min intensity display range value has the effect of thresholding the image it is applied to.

Because these images are in standard space we can turn on the atlas tools with Tools -> Toolbars -> Atlas Tools. Now as you click around in the image you can see reporting of the probability of being in different brain structures. The atlases that are turned on by default are the Harvard-Oxford cortical and subcortical atlases, but you can also choose others by pressing the Atlases... button. These are formed by averaging careful hand segmentations of structural images of many separate individuals. Find the thalamus (left or right); you can see that there is activation here.

You can turn on summary images of atlases by double-clicking on the name inside the Atlases... popup window; have a play with that. Finally, bring up the Structures... window, turn on the two tickboxes, and then click around in the list of structures. For those atlases which are probabilistic, this shows a single structure at a time, with the full probability information shown as a gradation of colour. The Structures... window is useful when you know the name of a brain area from a paper or talk, but you don’t know where it is anatomically.

Note that FSLview can also render 3D images of brains with activation shown. How to do this is documented in the online tutorial on the FSL website. Now kill fslview

## FSLUTILS

$ cd /storage/silver/pym0fm/<YOUR ITS LOGIN>/fsl\_course\_data/intro

### fslinfo

Type fslinfo highres and fslinfo thresh\_zstat1 and note their different image matrix dimensions (dim1-3 = X Y Z) and voxel sizes (pixdim1-3 = X Y Z).

Note that some images (highres) are of integer datatype, while others (thresh\_zstat1) are of floating point datatype. Integer means that the intensity values can only take on whole numbers - no fractions - raw image data is normally of this type. Floating point means that intensity values can be fractional - the result of applying most statistical processing algorithms to image data results in images of floating point type.

## BET (Brain Extraction Tool)

Most FSL programs can be run from the command-line without using a GUI, by typing fully lowercase names (e.g. bet). Most programs also have a GUI, which can be started via the fsl mini-GUI (type fsl &) or by typing a capitalised version of the command name (e.g. Bet &).

In any analysis, you should use the GUI to explore the data and try out analyses, but really, if you are going to do any serious analyses, you should write a script that will call out the tools you use.

For those FSL programs that can be run from the command line without a GUI the usage can be obtained by typing the lowercase name. Type bet and press return to see the usage for bet. This tells you that to use bet from the command line you need to give it the name of an input images, a filename for the output images, and you can also choose various options concerning the output and how brain extraction is achieved. Start the fsl GUI and click on BET, to see how this all works. Now exit both of these GUIs and start the Bet GUI directly from the terminal.

Using the Bet GUI, set the input file to structural; use the right-hand file selector rather than typing this in by hand. Turn on various optional outputs (brain extracted image, binary brain mask and skull surface image - see Advanced Options tab) but leave the other settings as they are. Note that the GUI suggests the default output name structural\_brain. When done click on "Go" to run BET and then exit the GUI once it's done.

Use ls to see what files got created and view the various outputs. Hint: if you type

$ ls -lrt

then the listing is sorted according to file creation date, hence it's very easy to see what the most recently created files in a directory are. View the brain mask overlaid onto the original image by first loading the original image into fslview and then adding the mask image. Change the transparency of the overlay so that you can get a better view of the success of the brain extraction.

Type bet to get the usage description again and make sure you understand the command that the GUI printed in the terminal when you pressed OK. Advanced use of BET to deal with problematic images is dealt with at the end of this document.

### Intra-subject Registration

An imaging session with a single subject usually results in several different kinds of images that have different dimensions (e.g. the structural has much smaller voxels than functional images, and therefore far more voxels). Intra-subject registration is the process of establishing a mapping between the two images, effectively “lining them up with each other”.

$ cd /storage/silver/pym0fm/<YOUR ITS LOGIN>/fsl\_course\_data/reg

We start by using FLIRT for image-to-image registration within a single subject. The images we start with in this example are of a different modality (T1-weighted and T2-weighted), and are called sub2\_t1 and sub2\_t2.

* Apply BET before FLIRT   
  Apply BET to the images sub2\_t1 and sub2\_t2 (saving the results as sub2\_t1\_brain and sub2\_t2\_brain). Check the outputs in FSLView - note that these do not need to be "perfect" but should have removed the majority of the non-brain structures without removing any of the brain material. It doesn't matter if there are some differences between them, especially around the eyeballs.
* Registration using the FLIRT GUI   
  Start the FLIRT GUI (using "Flirt &" or "Flirt\_gui &") and register sub2\_t2\_brain (input image) to sub2\_t1\_brain (reference image). Note that to make the file browser for the reference image go to the local directory (and not the directory containing the template images) just erase the contents of the "Filter" box (at the top) and press return.

Choose 6 DOF (which is not the default) since these images are from the same subject. 6 DOF implements a rigid body transform, made up of 3 directions of translation and 3 directions of rotation. Because the images are from the same subject you don’t want any expansion, contraction or shearing as part of the matching process. Choose Correlation Ratio as the cost function (which is the default) as the images are different modalities. (Mutual information is the most accurate, but it takes longer)

Save the output image as a name of your choice. This will take a few minutes to run...

* While you wait...   
  Load the input and reference volumes into FSLView (note that this is normally not possible as FSLView requires the two images to have the same resolution and FOV, which this pair just happens to have but it is not the norm). Once these are loaded, try to see the difference by flicking between them: that is, by turning the "top" one off and on (i.e. making them visible or not in the image list - by changing the "eye" symbol). Another way of comparing, which reduces the load on your visual short term memory, is to use the transparency slider to gradually make one of the images transparent, revealing the image underneath.
* Visual Check - FSLView   
  Once FLIRT has finished, add the resulting transformed image (output) to FSLView. Compare it to the reference image and the input image. Try to see the differences by flicking between pairs of images. Make sure that you only have two of them visible at any one time.

Flicking between images is a very good way of detecting registration differences and changes. Normally this can only be done for the reference and output images from FLIRT, as it is unusual that pre- and post- registered images share a common resolution and FOV like these do.

* Visual Check - slices   
  Now use slices to look at the result (the file output\_image or whatever you have called it instead) by running:

$ slices output\_image sub2\_t1\_brain

The red lines are edges generated from the second image (sub2\_t1\_brain) and should be aligned with edges in the output image, although sometimes they will show small differences in the BET masking around the outer brain surface. For comparison, also run slices on the reference image and the original image sub2\_t1\_brain and sub2\_t2\_brain. The misalignment is much greater at the back of the brain in this image pair.

### Inter-subject Registration

The objective of this practical is to try to register images of one subject to standard space with affine (FLIRT) methods. Non-linear (FNIRT) registration is covered in part 2 of this worksheet. We will work with an older subject that has slightly enlarged ventricles compared to the standard brain. This will illustrate the limitations of the affine (FLIRT) method. If you were dealing with such participants in your own research you might consider using non-linear (FNIRT) methods.

Note for users of SPM only: If you have used SPM you probably understand “inter-subject registration” to mean running co-registration between two participants as an alternative to separately normalising both brains. In the parlance of FSL, “inter-subject registration” means normalisation, which is actually the process of co-registration between a participant brain and the template brain. Anther big difference between FSL and SPM is that in FSL you don’t write a set of normalised functional images to the hard disc. Instead, only the final statistic images are actually written to the disc in templates space, and this only occurs by default when you run a 2nd level analysis. This will become clearer next week when you fit a GLM and run some contrasts.

* Registration command (FLIRT)   
  We will now use the command-line version of FLIRT, even though the GUI could also be used. With the command-line version of FLIRT we will register old\_subj to the standard space, but first we need to skull strip it. Run

$ bet old\_subj old\_brain -f 0.35

and take a look at the output to make sure it went to plan. Then we want to run FLIRT, linearly registering old\_subj to the standard space. We do this, using 12 DOF and the Correlation Ratio cost function, with the command

$ flirt -in old\_brain -ref $FSLDIR/data/standard/MNI152\_T1\_2mm\_brain -dof 12 -cost corratio -out flirted\_old\_brain -omat old\_subj\_to\_MNI152.mat

Note: this call must be typed as a single line, and the standard space image is not in the course data directory but can be found in $FSLDIR/data/standard (type "ls $FSLDIR/data/standard" to see the contents of this directory).

Note 2: $FSLDIR refers to a Linux “environment variable” that contains the name of the directory where FSL is installed. There are more useful Linux environment variables, such as $PATH. Type the name of a variable at the command prompt to see its value. You should also note that we used $FSLDIR/data/standard/MNI152\_T1\_2mm\_brain as our -ref, i.e. we register our skull-stripped brain to a skull-stripped template. To see the full range of command-line options just type "flirt". If this looks hard, you can also look at the command-line calls that the GUIs made in the previous examples.

* Visual check   
  Inspect the output image of your (affine) registration to see if it registered well to the reference image MNI152\_T1\_2mm\_brain. Toggle the visibility of one image on and off or use the transparency slider to do this. Note that affine registration to standard space is never "perfect", but should get major features such as the brainstem and brain outline quite close. Variation in the population is expected and is reflected in the blurring of structures within the standard (average) brain - although this standard brain was produced with non-linear registration and is less blurred than the affine version MNI152lin\_T1\_2mm. However, in this case the image is of an older subject and the enlarged ventricles will not fit perfectly with the standard brain.

Question: If a study comparing older subjects with younger subjects used affine registration to the template for both subject groups, how might this influence the results of the study?

* Transformation Matrix   
  When running FLIRT we also saved a transformation matrix which we called old\_subj\_to\_MNI152.mat (the argument to the -omat parameter). This is a text file containing a 4 by 4 array of numbers specifying the affine transform. The reason you selected the option for saving it to disc is because it is one of the required inputs for the non-linear FNIRT program, which is capable of fixing the ventricle problem when registering the older subject to the template. If you want to try this it is found in part 2 of this worksheet.

# Part 2: optional advanced FSL procedures

The following exercises are not essential to the course, but they provide good experience of using Linux, and they will also make you aware of some useful FSL utility programs for manipulating data and image files. You can do these if you have spare time in the workshop or in your own time.

$ cd /storage/silver/pym0fm/<YOUR ITS LOGIN>/fsl\_course\_data/intro

## fslsplit and fslmerge

run

$ fslsplit egfmri

to split up the 4D input image into the individual 3D images, or volumes, (vol0000, vol0001, vol0002, etc.). To list these new files do

$ ls vol\*

(note that the \* is expanded by the terminal to fit any characters in all possible filenames present, and so can be used in any commands, not just ls. \* is called a wildcard).

As an example let us remove the 3rd and 6th images from this sequence and then remerge the remaining ones. Remove the images by doing

$ rm vol0002\* vol0005\*

Check that these files have been deleted using looking at the directory contents using ls

Now remerge the remaining 3D images into a new 4D image with

$ fslmerge -t newfmri vol\*

The "-t" tells fslmerge to merge the images in time (you could also do other merges, such as stacking them on top of each other in space!) After "newfmri" (the name of the output image), fslmerge expects a list of images to be merged. For all FSL commands the image names can be used with or without including the filetype extension (extensions are .hdr .img .nii.gz etc.).

View the resulting image in fslview and check that there are only 8 volumes in the timeseries. You can also check the size of the timeseries with fslinfo. Also try concatenating in z instead of t and view in fslview; why does nothing happen when you try to turn on the movie loop?

Now tidy up by removing the redundant vol files using the rm command.

## fslroi

We will now fix an image that has somehow gotten "wrapped", with the nose sitting behind the back of the head. This sort of thing actually happens if you have a participant with a larger than average head and you don’t have enough “slice oversampling” in your protocol. The method used here might enable you to “recover” data that would otherwise be wasted.

$ fslview wrapped &

You will use fslroi to split this image into two (front and back parts) and fslmerge to remerge them back together in the right order. In FSLView, look at an inferior slice so that you can work out the y coordinate (in the anterior-posterior direction [= front to back]) where you want to make the split. Record the y-coordinate (in voxels, not mm).

Now use fslinfo to find out what the image matrix dimensions are (dim1,2,3). You're now ready to extract the back part of the head into a temporary file. If your y-coordinate was 76 you would do:

$ fslroi wrapped back 0 256 76 180 0 128

Now just type fslroi to see the usage. There are several ways to run fslroi: You can specify just a 3D ROI, as we have done here - this is the first of the usage possibilities. The second option is to extract a temporal ROI from a 4D dataset. The final option extracts a 4D ROI from a 4D dataset, controlling the region to be extracted in space and time. The program knows which option you want from the number of "arguments" that you type. Note that if you use the first option (3D ROI) on a 4D input file, the output will still be 4D, with the number of timepoints unchanged, and each volume cut down in 3D in the same way, as specified by the arguments.

Make sure that you understand the exact numbers used in the above command, that created back; note in particular that the second number in each pair is the length of the ROI in the relevant dimension, not the end point - the <ysize> was 180 (256-76) not 256.

Now work out how to run a similar command to create "front", from the back part of wrapped.

Finally, use fslmerge to merge front and back together correctly, and verify with fslview. Note that you could also have done a similar thing with the slight wraparound at the top of the image.

## More fun with BET

The following exercises illustrate more advanced use of BET to deal with brain extracting difficult images (e.g., where the image contrast is differs at the top and bottom of the head, or a lot of neck has been imaged)

Refer back to the previous section on BET and then do these exercises, which follow on from the earlier ones:

Try setting the Fractional intensity threshold (the -f option on the command line) to 0.2 and 0.8 in turn to see the effect it has (save to different output image names and load both into FSLView). This option controls the threshold that distinguishes brain from non-brain; if you reduce it from the default value of 0.5 then the brain estimate gets larger, and vice versa.

Run BET on sub3m0. Note that the cerebellum is underestimated. We can improve this by changing the gradient threshold -g (from its default of 0) to 0.5. This option causes the overall fractional threshold (controlled globally by -f) to vary linearly with slice number - to get smaller at the bottom and bigger at the top, or vice versa. Run this and note that we have now underestimated the top slices!

However by combining the gradient option with the main (-f) option we can get a good overall result: try -f 0.3 -g 0.2 (or maybe you can do even better!)

The final example is of a head image with a large amount of neck present - run BET on bighead to see a problem. Because of the neck, the initial brain surface is initialised too low in the image. There is an option on the command line to set the initial centre by hand; use fslview to get a rough guess (in voxels) of where the centre of the brain is in the image and see the improvement when you call bet with the -c option.

In some cases you can get BET to achieve the same improvement automatically by using the -R option; this re-runs BET several times, each starting its brain centre at the centre of the previously-estimated brain. Try the -R option instead of -c.

### Inter-subject Registration (non-linear)

* Registration command (FNIRT)

To supply non-linear FNIRT program with a good initial guess you will need to give it the transformation matrix output from the initial linear FLIRT step from part 1 of this workshop.

* Now we will look at non-linear registration of old\_subj to MNI152\_T1\_2mm. Note that for FNIRT we do not use the BET output nor do we register it to the skull stripped template, as it deals with non-brain tissue via a cost-function weighting mask in the standard space.

Running FNIRT takes around 20 minutes, so this has already been done using the following command (do not run this now):

$ fnirt --in=old\_subj --aff=old\_subj\_to\_MNI152.mat --config=T1\_2\_MNI152\_2mm.cnf  
--cout=coef\_old\_subj\_to\_MNI152 --iout=fnirted\_old\_subj --jout=jac\_old\_subj\_to\_MNI152 --jacrange=0.1,10

FNIRT actually takes a large number of parameters but here we have used the configuration file T1\_2\_MNI152\_2mm.cnf to specify most of these for us. The configuration file is simply a text file that contains many of the parameters that FNIRT takes (and values for them). This enables us to come up with a set of parameters that works well for matching a T1-weighted structural scan to the MNI152 T1-weighted standard brain and other common uses. You may see the parameters that were used by typing

$ cat $FSLDIR/etc/flirtsch/T1\_2\_MNI152\_2mm.cnf

The sight of that may convince you that configuration files are quite useful, and that unless you know exactly what you are doing you should use existing configurations. FSL comes with configuration files for doing things like matching T1 structural images to the MNI152 T1 standard brain (T1\_2\_MNI152\_2mm.cnf) and for matching FA-images to the FMRIB58 FA standard brain (FA\_2\_FMRIB58\_1mm.cnf). In the example above we have restricted the Jacobian-determinant ([see below](http://www.fmrib.ox.ac.uk/fslcourse/lectures/practicals/reg/index.html#jacobian)) to be in a more narrow range than is normally the case (default 0.01,100). We do this to facilitate calculating the inverse of the field [later](http://www.fmrib.ox.ac.uk/fslcourse/lectures/practicals/reg/index.html#inverse) in the practical.

Note that the FNIRT command requires that you specify the FLIRT matrix (old\_subj\_to\_MNI152.mat) that we generated previously. If you forget that you will most likely get poor results and may spend a long time trying to figure out why - like I did when preparing this practical. :-(

Also note that a registration always implies registering one image (typically -i or --in in FSL applications) to another. In the fnirt command above we can see that we want to register the image old\_subj, but to what? The --ref image is in this case specified in the configuration file and if you performed the cat command above you will have seen on the first row --ref=/usr/local/fsl/data/standard/MNI152\_T1\_2mm.

* Visual Check (images)   
  Load flirted\_old\_subj, fnirted\_old\_subj and MNI152\_T1\_2mm into fslview (Note that the last image can be added with "Add standard..."). Flip between the registered images and the template to see how well they match. It can also be interesting to flip back and forth between the linearly (flirted\_old\_subj) and the non-linearly (fnirted\_old\_subj) registered images. That may convince you that FNIRT has actually been quite gentle with your images, and on the whole the warps are relatively small and well behaved. It will appear as if the 2 images have a different smoothness, but ignore that for the now.
* Deformation Field   
  The output file coef\_old\_subj\_to\_MNI152 is not a normal image but is a special coefficient file format used by FNIRT, which is used to store the deformation field (also called a warpfield). It also contains the information that we passed from FLIRT to FNIRT (i.e. the information in old\_subj\_to\_MNI152.mat). You can view it (the coefficients) using fslview, but chances are that you will not feel much the wiser.

To view the deformation field itself it is necessary to convert the coefficients stored in coef\_old\_subj\_to\_MNI152 into fields. This is done using the command fnirtfileutils to change it into a vector image. If you run the command   
fnirtfileutils --in=coef\_old\_subj\_to\_MNI152 --ref=$FSLDIR/data/standard/MNI152\_T1\_2mm --out=field\_old\_subj\_to\_MNI152\_nonlinear\_only  
you will get a field (field\_old\_subj\_to\_MNI152\_nonlinear\_only) containing only the non-linear part of the warp, i.e. not including the affine part of the transform. This is useful for visual inspection since the non-linear part tends to be overshadowed by the affine component, making it difficult to interpret what one sees.

If instead you run the command   
fnirtfileutils --in=coef\_old\_subj\_to\_MNI152 --ref=$FSLDIR/data/standard/MNI152\_T1\_2mm --withaff --out=field\_old\_subj\_to\_MNI152  
the field (field\_old\_subj\_to\_MNI152) will now contain both the affine and non-linear components, which is necessary if we want to use them with e.g. convertwarp or invwarp.

* Visual Check (deformation field)   
  Now view the deformation field (use field\_old\_subj\_to\_MNI152\_nonlinear\_only for visual inspection) in fslview. Note that it is a 4D image, with three images in the fourth dimension. These correspond to x-displacements (Volume 0), y-displacements (Volume 1) and z-displacements (volume 2). You will often need to adjust the scaling of these images since fslview determines it from the first volume only. The values you observe when you click in the images are displacements (c.f. the little arrows in the lecture) in units of mm.

If you are doing well for time you can actually see "those arrows" by turning on "Display as Lines" in the "DTI display options" in fslview (from the "i" pop-up menu). They appear more clearly if you zoom in on a portion of the image. Although they are typically difficult to interpret, they can be quite decorative in a "haystacky" kind of way.

* Jacobian of deformation field (degree of expansion/compression)   
  Often more useful for viewing than the deformation fields themselves are the Jacobian determinants of the deformation fields. When we ran FNIRT we used --jout=jac\_old\_subj\_to\_MNI152 (see [above](http://www.fmrib.ox.ac.uk/fslcourse/lectures/practicals/reg/index.html#fnirt_command) if you have forgotten) to specify that we wanted the Jacobian determinants (one per voxel) written to the file jac\_old\_subj\_to\_MNI152.

Had we forgotten that we could still have generated the Jacobians at a later stage with the command

fnirtfileutils --in=coef\_old\_subj\_to\_MNI152 --ref=$FSLDIR/data/standard/MNI152\_T1\_2mm --jac=jac\_old\_subj\_to\_MNI152

The Jacobian field is useful for checking what liberties FNIRT has been taking with your data. The interpretation of the Jacobian is "voxelwise relative volume change" between the original (in our case old\_subj) and the warped (in our case fnirted\_old\_subj) image. A jacobian of 5 indicates that a volume in the original image has been shrunk by a factor of 5 (e.g. 1mm3 -> 0.2mm3). In general we don't want the Jacobian to be either too small or too large since it seems reasonable that a "function" that exists in both brains should be "implemented" in approximately equal amounts of brain tissue. To check this load jac\_old\_subj\_to\_MNI152 into fslview and click Tools->Image Histogram.

It (the Jacobian) is also useful in its own right, and is an important carrier of information e.g. in VBM. Look again at the Jacobian field and try to work out where the main volume changes has occurred. One area is the ventricle, which we expected. Can you see any other areas?

### If you are interested in using field maps to unwrap functional images using FUGUE, or registering partial brain images to templates see the FSL website for the necessary tutorials and software manuals.