PRACTICAL 3-1: PREPROCESSING I

PREPARING THE T1 IMAGE

Remove the Skull from the MP-RAGE

1. For many processing steps, including realignment, a skull stripped image is required. In FSL, this can be accomplished with the bet utility. Remember that a skull stripped image should be named by appending _brain to the end of the file name.

```
$ cd ~/data/TappingSubj2
$ bet MPRAGE_S2_RO MPRAGE_S2_RO_brain –f 0.2
```

2. Bet accepts many options which may be helpful for obtaining a good skull strip. An optimal skull strip removes all non-brain matter (including dura) while leaving all brain intact. This is hard to achieve with bet. For later processing steps, it is better to leave a bit of non-brain material than it is to remove brain.
3. Using FSL View to check the results of your attempts, try a few options with bet: (In checking the skull strips, you might want to try putting the _brain image on top of the original image in a different color with a transparency of about 0.5 to see which regions are removed by bet.)
    a. Try setting different fractional intensity thresholds with the –f flag. In IBIC we often use a value in the range of .05 to .15 for the MP-RAGE.
    b. Try giving a manual center-of-gravity with the –c flag. For this option, you should use FSL View to identify the voxel coordinates where the splenium meets the fornix along the midline and give those coordinates to bet.
    c. Try using the –R flag to repeat bet multiple times iteratively adjusting the center value. (This avoids you having to manually enter a set of center coordinates, but is likely not as effective as manually providing coordinates.)
    d. Try using the –B flag to remove the neck. This cannot be used with the –R flag, but in images with a neck, it is often preferable to use –B.
4. Make sure that your final image "MPRAGE_S2_RO_brain" reflects your best skull strip.

Run the MP-RAGE through FreeSurfer

1. FreeSurfer is a complete neuroimaging software suite, but it integrates nicely with FSL. We will barely touch on FreeSurfer's functionality, but structural processing in FreeSurfer starts with the recon-all command. Just by using recon-all, you can get a lot of potentially interesting information about the brain. Be aware that the recon-all process for a single subject takes several hours to complete. The following is an example of the command to run recon-all:

```
$ cd ~/data/
$ mkdir freesurfer
$ source /usr/local/freesurfer/stable5_3/SetUpFreeSurfer.sh
$ export SUBJECTS_DIR=~/data/freesurfer
$ recon-all –i ~/data/TappingSubj2/MPRAGE_S2_RO.nii.gz –subjid TappingSubj2 -all
```

Get Separate Tissue Masks from Fast

1. FreeSurfer provides nice segmentations of grey matter, white matter, and csf. We recommend using FreeSurfer's segmentations whenever possible. However, for ease of use in dealing with transformation matrices and saving time, we will get rougher versions of this information from FSL's fast tool for the purposes of this class.

```
$ cd ~/data/TappingSubj2
$ fast –t 1 –n 3 MPRAGE_S2_RO_brain &
```

2. When Fast finishes, you can use fslview to investigate the output. Which image is grey matter? White? CSF?

```
$ fslview MPRAGE_S2_RO_brain MPRAGE_S2_RO_brain_pve_*
```

PREPARING THE B0 IMAGES

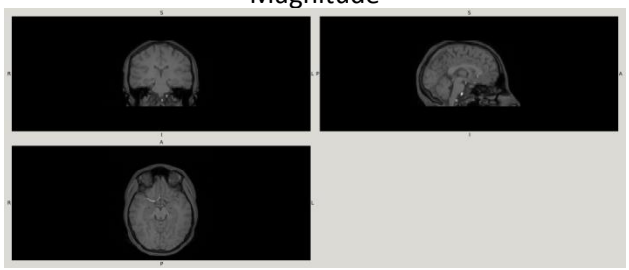
Identify the Phase and Magnitude Images

1. When EPI data are collected, they are distorted due to inhomogeneities (non-uniformities) in the B0 field. This signal distortion makes it hard to line the EPI images up with the structural images (particularly around air-tissue interfaces). We can collect a B0 (fieldmap) image to help with the registration of these two image types.
2. During conversion from PAR/REC to nifti, you generated two images named WIPB0_3mmISO_49slCLEARx1.nii.gz and WIPB0_3mmISO_49slCLEARx2.nii.gz. One of these images is a magnitude image and the other is a phase image. The magnitude image looks like a "traditional" MRI image in which you see a distinction between grey and white matter. The phase image is a map of the B0 field and looks the outline of the head in grey with dark or bright regions showing the location of distortion. Open the two B0 images to see which image is the magnitude and which is the phase.

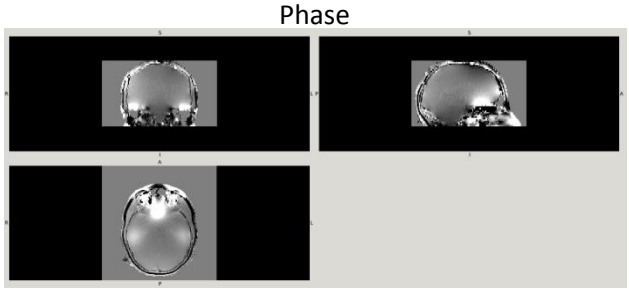
```
$ cd ~/data/TappingSubj2
$ fslview WIPB0* &
```

You will likely need to adjust the brightness in FSL View to see the phase image clearly.

Magnitude



Phase



3. Rename the images to indicate which is the magnitude image and which is the phase image.

```
$ mv WIPB0_3mmISO_49slCLEARx1.nii.gz B0_mag.nii.gz
$ mv WIPB0_3mmISO_49slCLEARx2.nii.gz B0_phase.nii.gz
```

Remove the Skull from the Magnitude Image

1. Removing the skull from B0 and functional images tends to be much less finicky than skull-stripping the MP-RAGE. Using the default Bet options usually works fine for the B0 magnitude image, but we will need a mask which is generated by the –m flag. Use fslmaths to dilate the brain mask to make it a bit bigger.

```
$ cd ~/data/TappingSubj2
$ bet B0_mag B0_mag_brain -m
$ fslmaths B0_mag_brain_mask -dilM B0_mag_brain_mask_dil
```

2. View the results of your skull strip in fslview.

```
$ fslview B0_mag B0_mag_brain B0_mag_brain_mask_dil
```

Unwrapping the Phase Image and Converting to Radians/second

1. The phase image on Philips scanners is written with units in Hertz. You can verify that the image is in Hertz by checking the range of values in the image.

```
$ fslstats B0_phase -R
```

An image in Hertz from the DISC scanner usually has a range of about -500 to 500. The range of your phase image should always be approximately symmetric around 0. If you notice that this is not the case, you may have a problem with scaling your image during conversion to nifti.

2. FSL's unwrapping program needs your phase image to be scaled from –Pi to Pi. Rescale your image by dividing by 500 and multiplying by Pi. Check the range of the new image to see the change.

```
$ fslmaths -dt float B0_phase -div 500 -mul 3.14 B0_phase_rescaled
$ fslstats B0_phase_rescaled -R
```

3. The prelude command will unwrap the phase image given the rescaled phase and skull-stripped magnitude images.

```
$ prelude -p B0_phase_rescaled -a B0_mag -o B0_phase_rescaled_unwrapped -m
B0_mag_brain_mask_dil
```

4. For further processing, the unwrapped phase image needs to be in radians/second. Use fslmaths to multiply the unwrapped phase image by 1000 to put it in radians/second.

```
$ fslmaths B0_phase_rescaled_unwrapped -mul 1000 B0_phase_rescaled_unwrapped_rads
```

PREPROCESSING THE FUNCTIONAL DATA

Functional data are messy – contaminated by all kinds of noise from motion, physiological processes, and scanner artifacts. Preprocessing helps to ready the data for the application of a model. In this section, we will cover fairly standard preprocessing steps. In the next section, we will cover additional options sometimes used to remove noise.

Motion Correction

1. Motion correction attempts to line up all of the volumes taken at different timepoints in a run with a single reference volume. Check the usage of mcflirt.

```
$ cd ~/data/TappingSubj2
$ mcflirt
```

Mcflirt takes several options, allowing you to control how the algorithm works (e.g., cost function, type of interpolation, volume used as reference) and the output you receive (e.g. transformation parameters that can be used as regressors, total displacement values).

2. We will run mcflirt to register all volumes to the middle volume in time, to provide transformation parameters and total displacement outputs, and to report its progress to the screen.

```
$ mcflirt -plots -rmsrel -rmsabs -report
-in Resting
```

3. Check your directory contents to see that mcflirt has created several files.

```
$ ls Resting*
```

4. Resting_mcf.nii.gz is the motion corrected image file. Compare the motion corrected file to the raw file. In movie mode, the brain shouldn't move around following motion correction. If the brain still moves, you might consider excluding specific volumes or the whole run from data analysis.

```
$ fslview Resting.nii.gz
Resting_mcf.nii.gz
```

5. Resting_mcf.par is the motion parameter file. It contains 6 columns of data – the first 3 for each rotational movement (in radians around x, y, and z); the last 3 for each translational movement (in mm in the x,y, and z directions). Check the contents of this file.

```
$ cat Resting_mcf.par
```

Each column contains 1 row for every timepoint in your functional data and represents the amount that the

FAQ: Which preprocessing steps should I apply to my data?

It should be noted that there's a huge amount of discussion in the literature about which preprocessing steps should be applied and in which order. Many of these decisions are study-specific as the right choice may depend on aspects of the design of your study (e.g., the TR, the brain regions of most interest to you, whether the data are taken during a task or at rest). Some authors have even suggested that the optimal processing pipeline may be different for every participant in the same study. Choices made in preprocessing can have large effects on your final results; when you set up a processing pipeline, it is important to put thought into your preprocessing and analysis methods and to consult with others who are familiar with the tradeoffs of each option.

volume was moved in that direction to match the reference volume. This parameter file can be fed into other programs as "motion regressors." It is important to note that these values are not a measurement of how much the participant moved. They are measures of how much mcflirt adjusted the volume. (See that the values for the reference volume are all 0.) If mcflirt was not able to appropriately register the functional run, the motion correction parameters will be a poor estimate of actual subject movement.

6. Resting_mcf_abs.rms is an estimate of total displacement in mm (across all 6 dimensions) for each volume relative to the reference volume. Resting_mcf_abs_mean.rms is the mean of these displacement values across time. Check the contents of these files.

```
$ cat Resting_mcf_abs.rms
$ cat Resting_mcf_abs_mean.rms
```

7. Resting_mcf_rel.rms is an estimate of total displacement in mm (across all 6 dimensions) for each volume relative to the previous volume. Resting_mcf_rel_mean.rms is the mean of these displacement values across time. Check the contents of these files.

```
$ cat Resting_mcf_rel.rms
$ cat Resting_mcf_rel_mean.rms
```

8. Total displacement values (either absolute or relative) are often used to decide whether a volume or run of data should be excluded for excessive motion. How much motion is too much is a matter of debate, but a threshold of ½ a voxel or 1 voxel (in this case, 3mm) for any volume is fairly common.

Slice Timing Correction

1. Within a single volume, slices of the volume are acquired one at a time. At DISC, axial slices are often acquired sequentially from the bottom of the brain to the top. As a result, we capture activity in different parts of the brain at different times relative to the onset of a stimulus. We can correct for these slice timing differences in fsl using slicetimer to correct the data so that it will be close to whatever it would have been if all slices were acquired simultaneously. With short TRs (on the order of 2 seconds), slice timing errors are likely to be very small and many people choose not to apply a slice timing correction. Check the usage of slicetimer.

```
$ slicetimer
```

2. Run slicetimer on the motion corrected data with the correct TR (in this case 2.5 seconds). The default order of acquisition (bottom-up sequential) is correct here. Otherwise, we would need to use flags like –down and –odd or a custom slice order (--ocustom).

```
$ slicetimer –i Resting_mcf –r 2.5
```

3. The output file is called Resting_mcf_st.nii.gz. Use FSL View to check this file.

Temporal Filtering

1. Temporal filtering allows us to remove noise from our data when the noise we want to remove has a different frequency than that of the signal we are interested in. High-pass filters pass high frequencies – that is they stop low frequencies from getting through the filter, so they can be used to remove low frequency noise. Low-pass filters pass low frequencies, so they can be used to remove high frequency noise.
   If both a high-pass filter and a low-pass filter are applied, it is called band-pass filtering, because only signals within a certain frequency band are allowed through the filter.
   High-pass filters are commonly used in fMRI as they can remove known noise sources (e.g., scanner drift). Low-pass filters are more controversial, but they are sometimes employed in the processing of resting state data. If you are very interested in changes across time (e.g., dynamic properties of resting state data), it is likely best to avoid any temporal filtering.
2. Filtering can be achieved in FSL with fslmaths. Check the usage of fslmaths to see the syntax for bandpass filtering.

```
$ fslmaths
```

3. Fslmaths requires a sigma value for the high-pass filter and one for the low-pass filter. We are only going to apply a high-pass filter. To skip the low-pass filter, we will use a -1 in place of the low-pass sigma. Calculation of sigma can be a bit tricky.
    a. If we want to use a high-pass filter to retain frequencies faster than 0.01 Hz, we first must convert this value to seconds. Hertz are cycles/second, so taking the reciprocal of 0.01 gives a filter of 100 seconds.
    b. To get this filter value into volumes, divide the value in seconds by the length of the TR (2.5 seconds) to get 100/2.5=40.
    c. To get sigma (the half-width of the filter), divide this filter in volumes by 2 to get 40/2=20. Setting the low-pass value of the band-pass filter to -1 means all frequencies are allowed through the low-pass filter (so there is no low-pass filter).

```
$ fslmaths Resting_mcf_st –bptf 20 -1 Resting_mcf_st_tempfilt
```

Spatial Smoothing

1. Spatial smoothing lowers the spatial frequency of the image in order to improve your signal to noise ratio. It can also help with imprecisions in realignment to standard space across subjects. The size of a smoothing kernel varies across studies, but the FWHM is often on the order of twice the voxel dimension (e.g., 6mm FWHM for a 3mm isotropic voxel). A smaller size may be chosen when increased precision is desirable (e.g., for MVPA).
2. Spatial smoothing in FSL can be applied using susan. Check the usage of susan.

```
$ susan
```

3. We will use susan to apply a 3D smoothing kernel with a half-width of 3mm (FWHM=6mm) with the default brightness threshold and a median filter applied where single-point noise is detected.

```
$ susan Resting_mcf_st_tempfilt -1.0 3 3 1 0 Resting_mcf_st_tempfilt_smooth
```

Functional to Structural Registration

1. Registration is not a preprocessing stepcd per se as we generally avoid interpolations associated with moving the functional data between spaces. Rather, statistical images or masks are usually moved between spaces. Nonetheless, it can be quite useful to estimate registration matrices during preprocessing so these matrices are available later in processing/analysis.
2. To register functional data to the T1 image, we will use epi_reg. Check the usage of epi_reg

```
$ cd ~/data/TappingSubj2
$ epi_reg
```

Note that epi_reg can be used with or without a field map. Epi_reg is a very memory intensive process. It will take a while and you may not be able to use your computer for anything else while it is running. In practice, you may want to start this running on a remote machine or before you go to a meeting/leave for the day.

3. For registration with the functional image, we will choose a representative functional volume. Since mcflirt realigned all of our images to the middle volume, we will use the middle volume as a reference here. Find the middle volume and extract it. Then, skull-strip the extracted volume for registration.

```
$ fslval Resting_mcf_st_tempfilt_smooth dim4
$ fslroi Resting_mcf_st_tempfilt_smooth Resting_midvol 72 1
$ bet Resting_midvol Resting_midvol_brain
```

4. To use epi_reg without a field map, you only need the epi image that you just created and the T1 MP-RAGE. You will need to have a skull-stripped version of the T1.

```
$ epi_reg --epi=Resting_midvol_brain --t1=MPRAGE_S2_RO
--t1brain=MPRAGE_S2_RO_brain --out=rest2t1 &
```

5. To use epi_reg with a field map, you also need the skull-stripped B0 magnitude image, the B0 phase image in radians/s, the echo spacing, and the phase encoding direction. You previously created all of the necessary images, but will need to determine the echo spacing and phase encoding direction.

a. Echo spacing: The echo spacing is calculated based on the water fat shift and the echo train length (EPI factor). You can find those values in the .PAR file for the functional run.

```
$ gedit parrec/TappingSubj2_WIP_Resting_SENSE_5_1.PAR
```

The formula for calculating echo spacing is Spacing = (1000*WFS)/(434.215*(EPIFACTOR + 1)) . You can also get this value with less work using the following script:

```
$ /NEUROCOURSE/Templates/Programs/ibicB0Factors parrec/TappingSubj2_WIP_Resting*.PAR
```

As noted in the usage for epi_reg, if acceleration is used during acquisition, the echo spacing should be divided by the acceleration factor. In these data, the SENSE acceleration factor was 2, so the effective echo spacing is actually 0.63/2=0.315 ms or 0.000315 seconds.

b. Phase encoding direction: The axis for phase encoding can be found in the .PAR file (Preparation direction). For the 3T Philips Achieva at DISC, you should choose unwarping direction "y". You can check this by trying correction in both directions (y and –y) for a given dimension. Using the proper phase encoding direction should make your registration results better.

```
$ epi_reg --epi=Resting_midvol_brain --t1=MPRAGE_S2_RO
--t1brain=MPRAGE_S2_RO_brain --fmap=B0_phase_rescaled_unwrapped_rads
--fmapmag=B0_mag --fmapmagbrain=B0_mag_brain --echospacing=0.000315 --pedir=y
--out=rest2t1_w_B0
```

6. To view the output of registration, use fslview to open the output images along with the T1 brain.

```
$ fslview rest2t1 rest2t1_w_B0 MPRAGE_S2_RO_brain
```