Etkin Lab's Analysis Pipeline User's guide

Brian Patenaude

January 22, 2015

Chapter 1

Overview and Setup

1.1 Introduction

Our *analysis pipeline* is merely a bash script that bring together our structural and functional processing stream. It brings together components of FSL and SPM as well as integrates some custom code (typically implemented in C++).

1.2 Software requirements.

This pipeline is for use on OSX or linux (tested on CentOS/RedHat). It requires the installation of FSL and SPM8. There will be some adjustments to the SPM8 configuration for our purposes. For OSX, it requires a version of readlink that is consistent with linux (i.e. has the -f option).

1.3 Installation

1.3.1 Getting the data

If you are reading this, you should already have access to the *analysis_pipeline* git repository that is hosted on `www.bitbucket.org`. The first step is to clone the repository into a local folder. The location is up to you, however, if you are planning to parallelize the jobs on a grid, it should by copied to a centralized location. The follow command will clone the repository,

```
git clone https://bitbucket.org/bmpatena/analysis_pipeline
```

This will create a directory with an assortment of files in it needed for you to run the pipeline. Now it's time to setup your environment variables.

The file spm_defaults.m should be used instead of that distributed by spm. There is two differences: 1) disables the implicit masking (i.e. set threshold to -inf) and 2) defaults.cmdline=true which is used to disable the GUI. The latter is required in order to parallelize/run in background. This file needs to substitute that which is in SPM.

1.3.2 Setting Up Environment

It is assumed at this point that FSL has been installed and you environment has been setup. I've set these environments computer wide by editing /etc/profile. Otherwise, you can add to you personal profile. From a lab management perspective, the former facilitate consistency across all users.

```
#location of the install
export ANALYSIS_PIPE_DIR=/PATH_TO_SRC/analysis_pipeline/
export PATH=$ANALYSIS_PIPE_DIR:$PATH

#location of SPM install used for pipeline.
export SPM8DIR=/Applications/spm8_sge
#fink
source /sw/bin/init.sh
```

Some Notes

- I use a local copy of SPM for performance issues with network copy; may not be an issue for you.

- SPM image IO is not very friendly with network file system, this may be a cause of slow down if too many parallel instances exist.
- *source /sw/bin/init.sh* is only need for OSX. fink is used to install a version of *readlink* that is consistent with linux.

Chapter 2

Quick Start Guide

2.1 Common options

The basic options are those which I've found that most people use most frequently. They've been taken from scripts used from analyses that we have performed. For clarity I've used the image extensions in the examples, but they are not necessary.

-func_data *func_4D* : Proceeded by the 4D functional data (EPI or spiral).

-t1 *im_t1* : Proceeded by the highres structural image (T1 weighted).

-reg_info : Optional, specified to use existing structural analysis folder. Proceeded by the structural analysis directory.

-design : Proceeded by a Matlab .mat file of the design matrix. The internal structure is that specified by SPM.

-spm_contrast : Proceeded by an SPM contrast file (.m file).

-output_extension *Analysis* : Proceed by the extension that will be used for the output. It combined the name specified by **-func_data** and append a "." plus whatever extensions. "+" characters will be prepended in the case the directory exists. e.g *func_4D.Analysis*.

-model_name *ModelName* : Proceed by a name. A folder, *ModelName.spm* will be created in the output directory, this contains the final SPM analysis.

-motion : No arguments. This options indicates to the pipeline to include motion regressors first level model.

-tr : Proceeded by a number. The number is the TR from the acquisition sequence in seconds(time between time points).

-deleteVolumes : Proceeded by an integer. The number of volumes to be deleted from the beginning of the time series.

2.2 Running Structural Analysis

I typically run the structural analyses as a separate stage. This is done for 2 reasons: 1) To be able to QC the registration prior to proceeding with first level models. 2) With multiple functional tasks, each can just reference (and link to) this analysis. This saves a lot of computation time. Before running, please see the notes on orientation below. Orientation of the images are assumed to be handled in advance and is not accounted for in these scripts.

To run the structural analysis,

```
analysis_pipeline.sh -struct_only -t1 subjectID_struct_t1.nii.gz  -output_extension struct_only
```

2.2.1 Notes on Orientation

Generally speaking the orientation should be that which matches the MNI template. This is the common orientation used by FSL and SPM. This sometimes differs from what's output by DICOM converters. For example, **Freesurfer**'s *mri_convert* tool. To re-orient the image I use *fslreorient2std*, this does require properly set NIFTI headers. Note that the orientation labels for an image will be displayed along side the image in **FSLView**.

Radiological vs Neurological

PLEASE BE VERY CAREFUL with left/right orientation. Although, to my knowledge, FSL handles the Radiological vs Neurological orientation internally and is accounted for in all tools, this may not be the case for all tools. By default, we convert everything into Radiological format upon reconstruction. ***CAUTION should be take when doing this since it may not be evident when done incorrectly. Refer to `http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/Fslutils` for more details.

2.2.2 Checking your data

The following commands assume that you are in the analysis directory. I use FSLView to overlay and check images. Let's first check to see if the brain extraction worked.

```
fslview struct/orig struct/brain_fnirt
```

If you'd like to view the difference FNIRT adds to the brain mask you can overlay both brains,

```
fslview struct/orig  struct/brain struct/brain_fnirt
```

Let's check the structural to MNI space registration,

```
fslview data/standard/MNI152_T1_2mm  reg/highres2standard_warped.nii.gz
```

Checking tissue segmentation,

```
fslview struct/orig  struct/brain_fnirt_seg.nii.gz
```

Checking subcortical segmentation,

```
fslview struct/orig  struct/first_all_fast_firstseg.nii.gz
```

2.3 Run a basic first level analysis

```
analysis_pipeline.sh  -func_data <fmri_4D.nii.gz>   -t1 <subjectID_struct_t1.nii.gz>  \
        -reg_info <subjectID_struct_t1.struct_only> -design <spm_design_matrix.mat> \
        -spm_contrast  <spm_contrast_file.m> -model_name <name_of_model> \
        -output_extension <analysis_directory_extension>  -motion -tr <TR> \
        -deleteVolumes <Number_of_Volumes_to_Delete>
```

Some Notes

- The SPM design matrix should already account for the volumes to be deleted.

Chapter 3

Guide to Data Structure and Files

The data structure mostly follows that laid out in FSL with some obvious differences due to our incorporation of SPM. The following section are separated by folder that are created in the analysis directory.

struct : Structural processing information.

reg : Registration information.

model.spm : First level SPM models (native functional space).

reg_standard : Contrast images transformed into standard space.

3.1 struct

orig.nii.gz : Copy of the original structural image.

Produced by first_flirt :

first_flirt.mat : Estimate of linear transform for **FIRST**'s subcortical models.

first_flirt.nii.gz : *orig.nii.gz* transformed to 1mm MNI space using *first_flirt.mat*

first_flirt_cort.mat :

first_flirt_cort.nii.gz :

first_flirt_cort_inv.mat :

first_flirt_tmp_cort_stage2.mat :

Using FLIRT to transform MNI space brain mask, *mni_bet_1mm_mask.nii.gz*, to structural space and apply to images using **fslmaths**:

brain_mask.nii.gz : Transformed brain mask in structural space.

brain.nii.gz : An initial estimate of the brain (i.e. brain extraction) using **first_flirt** and a standard space MNI mask.

Produced using FNIRT, with *orig.nii.gz* and *brain.nii.gz* as input.

brain_fnirt.nii.gz : A refined estimate of the brain voxels using **FNIRT**.

brain_fnirt_bias.nii.gz : Bias field estimate produce by **FNIRT**.

brain_fnirt_mask.nii.gz : Binary mask of the estimate brain voxels (propagating standard space mask via **FNIRT**).

orig_to_MNI152_T1_2mm.log :

Applying FAST to *brain_fnirt.nii.gz* produces,

brain_fnirt_mixeltype.nii.gz :

brain_fnirt_pve_0.nii.gz :

brain_fnirt_pve_1.nii.gz :

brain_fnirt_pve_2.nii.gz :

brain_fnirt_pveseg.nii.gz :

brain_fnirt_seg.nii.gz :

brain_fnirt_wmseg.nii.gz :

Applying FIRST to *orig.nii.gz* produces,

first : A directory containing individual model fits

first_all_fast_firstseg.nii.gz : The combined subcortical segmentation image