EPItome-xl

As long as our brain is a mystery, the universe, the reflection of the structure of the brain will also be a mystery.

SANTIAGO RAMÓN Y CAJAL

Joseph D. Viviano
Department of Biology
York University
Toronto, ON, CA
joseph.d.viviano@gmail.com

Contents

1 Introduction

EPItome-xl is a program designed for the flexible construction of MRI pre-processing pipelines, with a focus on functional MRI images and their associated problems. Its primary function is to take BASH modules and chain them together in any way the user desires to create a set of batch-processing scripts for an MRI experiment. These modules are not necessarily dependent on one another, allowing users of this package to easily extend the functionality of EPItome-xl by simply depositing a shell script into the appropriate module folder and writing the associated python wrapper command (and documentation!)

The goal of this design is to allow multiple levels of control for users of different skills with the same interface. EPItome facilitates the construction of very robust pre-processing scripts that can be run on your computer or in a distributed computing environment by only answering a few high-level questions, hopefully making it easy for beginners to get started. The scripts that are output by these commands are otherwise fully tweak-able and well commented – they encourage experimentation. These modified modules could eventually evolve into new features altogether, which are easily added to the existing pool.

This system is also designed to facilitate easy-to-reproduce research, as these scripts can be easily re-purposed for new experiments that follow the EPItome folder structure. In this way, the outputs of EPItome act as your lab notebook, and can be shared with collaborators or reviewers.

This manual will progress to more advanced topics in the end. First, I will explain the basic use of EPItome-xl. Next, I'll walk the user through a few common pre-processing tasks. I'll then explain the modules one-by-one, and finish with an explanation on how to add new modules.

2 Requirements & Dependencies

EPItome contains a small number of programs that actually analyze data, but also makes heavy use of widely-used MRI analysis tools and a number of python distributions. The user is assumed to have properly installed and configured FSL, AFNI, Freesurfer, and the python packages numpy, scipy, and matplotlib.

The program itself was built and tested on a Ubuntu 12.04 server. I imagine it will work well in any linux environment. It should run on Mac OS X as well, but this remains unverified. There will be no official support for Windows.

3 Using EPItome-xl

After installation, EPItome requires you to specify a few paths. These settings can be found in the /epitome/config.py file.

- dir_data: should point to your MRI experiment folder.
- dir_pipe: should point to your installation of EPItome-xl.
- dir_afni: should point to your installation of AFNI.
- cores: could be set to the maximum number of cores you wish EPItome to use (defaults to the maximum possible -1).

EPItome-xl comes with a few command-line interfaces. EPItome is used to inspect data in the MRI directory, returns information on the currently-available modules, can be used to construct new pipelines, and to remove unwanted data from the MRI directory cleanly. EPIphysio is a tool built to parse physiological data from the BIOPAK 150 unit we have installed at York University (Toronto), and might need to be adapted / extended to work with other units. EPIfolder is used to generate an appropriate folder structure in the MRI directory for the EPItome pipeline to work on.

The MRI directory itself must be organized as follows:

```
/
├── EXPERIMENTS
│   └── SUBJECTS
│       └── MODE
│           └── SESS
│               └── RUN
└── FREESURFER
    └── SUBJECTS
```

The Freesurfer subject directory should point to the Freesurfer directory in your MRI folder – this can easily be accomplished using a link. The remaining folder structure is perhaps best generated by using the included EPIfolder tool (or via a home-brew script).

3.0.1 EXPERIMENTS

3.0.2 SUBJECTS

3.0.3 MODE

3.0.4 SESS

3.0.5 RUN

4 The Pipeline

For the pipeline to work, your RAW data must be converted to the appropriate format in the WORKING directory. EPItome-xl should be installed and configured properly on your server (code & up-to-date installation instructions are available from GitHub).

What follows is a description of what each module in EPItome-xl does. These modules can be easily chained together manually, or by using the command-line interface included with the pipeline. New modules are simply bash scripts which call various programs, including FSL, Freesurfer, AFNI, and custom python programs. Therefore, functionality of EPItome is easily extended without perturbing the function of older modules.

The types of modules included can be roughly split into 4 categories: freesurfer, pre-processing, quality-control, and cleanup.

4.1 Freesurfer

Right now, the default freesurfer recon-all is run on every participant before further processing. This is to produce surface files that can be used for cortical smoothing / data visualization, and the automatic generation of tissue masks which can be used for the generation of nuisance regressors.

4.1.1 fsrecon.py

Usage: `fsrecon.py <data_directory> <experiment> <modality> <cores>`

data_directory – full path to your MRI/WORKING directory.
experiment – name of the experiment being analyzed.
modality – image modality to import (normally T1).
cores – number of cores to dedicate (one core per run).

This sends each subject's T1s through the Freesurfer pipeline. It uses multiple T1s per imaging session, but does not combine them between sessions. Data is output to the dedicated FREESURFER directory, and should be exported to the MRI analysis folders using `fsexport.py`.

4.1.2 fsexport.py

Usage: `fsexport.py <data_directory> <experiment>`

> data_directory – full path to your MRI/WORKING directory.
> experiment – name of the experiment being analyzed.

Imports processed T1s from Freesurfer to the experiment directory.

4.2 Pre-Processing

This contains the lion's share of the pipeline. Every run of EPItome begins with `init_EPI`, which contains a non-contentious set of pre-processing steps for EPI images. THe following stages can be chained together at will to preform de-noising, spatial transformations, projections to surface-space, and spatial smoothing.

4.2.1 init_EPI

Usage: `init_EPI <data_quality> <del_tr> <t_pattern> <normalization> <masking>`

> data_quality – 'low' for poor internal contrast, otherwise 'high'.
> del_tr – number of TRs to remove from the beginning of the run.
> t_pattern – slice-timing at acquisition (from AFNI's 3dTshift). normalization – voxel wise time series normalization. One of 'zscore', 'pct', 'demean'.
> masking – EPI brain masking tolerance. One of 'loose', 'normal' 'tight'.

Works from the raw data in each RUN folder. It performs general pre-processing for all fMRI data:

- Orients data to RAI
- Deletes initial time points (optionally)
- Removes data outliers

- Slice time correction
- Deobliques & motion corrects data
- Creates session mean deskulled EPIs and whole-brain masks
- Scales and optionally normalizes each time series
- Calculates various statistics + time series

Times series normalization can be accomplished in one of two ways: percent signal change, and scaling. For percent signal change, the data is normalized by the mean of each time series to mean = 100. A deviation of 1 from this mean value indicates 1% signal change in the data. This is helpful for analyzing only relative fluctuations in the signal and is best at removing inter-session/subject/run variability, although it can also introduce rare artifacts in small localized regions of the images and may not play well with multivariate techniques such as partial least squares without accounting for these artifacts. Alternatively, one can scale the data, which applies single scaling factor to all voxels such that the global mean of the entire run = 1000. This will help normalize baseline shifts across sessions, runs, and participants. Your selection here might be motivated by personal preference, or in rarer cases, analytic requirements. When in doubt, it is safe to select 'off', as scaling can be done later by hand, or 'scale' if one is doing a simple GLM-style analysis. 'pct' should be used by those with a good reason.

Masking options are provided to improve masking performance across various acquisition types, but it is very hard to devise a simple one-size fits all solution for this option. Therefore the QC outputs will be very important for ensuring good masking, and these options may need to be tweaked on a site-by-site basis. Luckily, many analysis methods do not rely heavily on mask accuracy. In cases that do, such as partial least squares / ICA / PCA analysis, close attention should be paid to the output of this step. Hopefully the 'loose', 'normal', and 'tight' nomenclature are self-explanatory. Generally, it is best to start with normal, and adjust if required.

Prerequisites: None.

4.2.2 combine_volumes

Usage: `combine_volumes <func1_prefix> <func2_prefix>`

func1_prefix – functional data prefix (eg., smooth in func_smooth).
func2_prefix – functional data prefix (eg., smooth in func_smooth).

Combines two functional files via addition. Intended to combine the outputs of `surfsmooth` & `surf2vol` with `volsmooth`, but could be used to combine other things as well. The functional files should not have non-zeroed regions that overlap, or the output won't make much sense.

Prerequisites: Two EPI modules with unique output prefixes. Intended to be used to combine the outputs of `volsmooth` and `surfsmooth` in a single volume.

4.2.3 linreg_calc_AFNI

Usage: `linreg_calc_AFNI <cost> <reg_dof> <data_quality>`

> cost – cost function minimized during registration.
> reg_dof – 'big_move' or 'giant_move' (from align_EPI_anat.py).
> data_quality – 'low' for poor internal contrast, otherwise 'high'.

Uses AFNI's align_EPI_anat.py to calculate linear registration between EPI <–> T1 <–> MNI152, and generate an EPI template registered to T1 & T1 registered to EPI (sessionwise). Specific options can be found in the command-line interface's help function.

Prerequisites: `init_EPI`.

4.2.4 linreg_calc_FSL

Usage: `linreg_calc_FSL <cost> <reg_dof> <data_quality>`

> cost – cost function minimized during registration (see FSL FLIRT).
> reg_dof – 6, 7, 9, or 12 degrees of freedom (see FSL FLIRT),
> data_quality – 'low' for poor internal contrast, otherwise 'high'.

Uses FSL's FLIRT to calculate linear registration between EPI <–> T1 <–> MNI152, and generate an EPI template registered to T1 & T1 registered to EPI (sessionwise). Specific options can be found in the command-line interface's help function.

Prerequisites: `init_EPI`.

4.2.5 linreg_EPI2MNI_AFNI

Usage: `linreg_EPI2MNI_AFNI <func_prefix> <voxel_dims>`

> func_prefix – functional data prefix (eg.,smooth in func_smooth).
> voxel_dims – target voxel dimensions (isotropic).

Prepares data for analysis in MNI standard space.
Prerequisites: `init_EPI`, `linreg_calc_AFNI`.

4.2.6 linreg_EPI2MNI_FSL

Usage: `linreg_EPI2MNI_FSL <func_prefix> <voxel_dims>`

func_prefix – functional data prefix (eg., smooth in func_smooth).
voxel_dims – target voxel dimensions (isotropic).

Prepares data for analysis in MNI standard space.
Prerequisites: `init_EPI`, `linreg_calc_FSL`.

4.2.7 linreg_FS2EPI_AFNI

Usage: `linreg_FS2EPI_AFNI`
Brings Freesurfer atlases in register with single-subject EPIs.
Prerequisites: `init_EPI`, `linreg_calc_AFNI`.

4.2.8 linreg_FS2EPI_FSL

Usage: `linreg_FS2EPI_FSL`
Brings Freesurfer atlases in register with single-subject EPIs.
Prerequisites: `init_EPI`, `linreg_calc_FSL`.

4.2.9 linreg_FS2MNI_FSL

Usage: `linreg_FS2MNI_FSL`
Brings Freesurfer atlases in register with MNI standard space.
Prerequisites: `init_EPI`, `linreg_calc_FSL`.

4.2.10 gen_regressors

Usage: `gen_regressors <func_prefix>`

func_prefix – functional data prefix (eg.,smooth in func_smooth).

Creates a series of regressors from fMRI data and a freesurfer segmentation:

- white matter + eroded mask
- ventricles + eroded mask

- grey matter mask
- brain stem mask
- dialated whole-brain mask
- draining vessels mask
- local white matter regressors + 1 temporal lag
- ventricle regressors + 1 temporal lag
- draining vessel regressors + 1 temporal lag

Prerequisites: `init_EPI`, `linreg_calc_AFNI/FSL`, `linreg_FS2EPI_AFNI/FSL`.

4.2.11 gen_gcor

Usage: `gen_gcor <func_prefix>`

func_prefix – functional data prefix (eg.,smooth in func_smooth).

Calls an AFNI script to calculate the global correlation for each concatenated set of runs (across all sessions). Useful for resting state functional connectivity experiments.

Prerequisites: `init_EPI`.

4.2.12 filter

Usage: `filter <func_prefix> <det> <gs> <vent> <dv> <wm_loc> <wm_glo>`

func_prefix – functional data prefix (eg.,smooth in func_smooth).
det – polynomial order to detrend each voxel against.
gs – if == on, regress mean global signal from each voxel.
vent – if == on, regress mean ventricle signal from each voxel.
dv – if == on, regress mean draining vessel signal from each voxel.
wm_loc – if == on, regress local white matter from target voxels.
wm_glo – if == on, regress global white matter for all voxels.

This computes detrended nuisance time series, fits each run with a computed noise model, and subtracts the fit. Computes temporal SNR. This program always regresses the motion parameters & their first lags, as well as physiological noise regressors generated my McRetroTS if they are available. The rest are optional, and generally advisable save global mean regression.

Prerequisites: `init_EPI`, `linreg_calc_AFNI/FSL`, `linreg_FS2EPI_AFNI/FSL`, `gen_regressors`.

4.2.13 TRdrop

Usage: `TRdrop <func_prefix> <head_size> <FD_thresh> <DV_thresh>`

> func_prefix – functional data prefix (eg.,smooth in func_smooth).
> head_size – head radius in mm (def. 50 mm).
> thresh_FD – censor TRs with Δ motion > x mm (def. 0.3 mm).
> thresh_DV – censor TRs with Δ GS change > x % (def. 1000000 %).

This removes motion-corrupted TRs from fMRI scans and outputs shortened versions for connectivity analysis (mostly). By default, DVARS regression is set of OFF by using a very, very high threshold.

Prerequisites: `init_EPI`.

4.2.14 lowpass

Usage: `lowpass <func_prefix> <mask_prefix> <filter> <cutoff>`

> func_prefix – functional data prefix (eg.,smooth in func_smooth).
> mask_prefix – mask data prefix (eg., EPI_mask in anat_EPI_mask).
> filter – filter type: 'median', 'average', 'kaiser', or 'butterworth'.
> cutoff – filter cuttoff: either window length, or cutoff frequency.

This low-passes input data using the specified filter type and cutoff.

Both 'median' and 'average' filters operate in the time domain and therefore, the best cutoff values are odd (and must be larger than 1 to do anything). Time-domain filters are very good at removing high-frequency noise from the data without introducing any phase-shifts or ringing into the time series. When in doubt, a moving average filter with window length of 3 is a decent and conservative choice.

Alternatively, the 'kaiser' and 'butterworth' filters work in the frequency domain and accept a cutoff in Hz (people tend to use a default of 0.1). Both are implemented as bi-directional FIR filters. The kaiser window is high order and permits reasonably sharp rolloff with minimal passband ringing for shorter fMRI time series. The butterworth filter is of low order and achieves minimal passband ringing at the expense of passband roll off (in layman's terms, butterworth filters will retain more high-frequency content than a kaiser filter with equivalent cutoff). The effect of the passband ringing is an empirical question that would be best tested by the User.

Prerequisites: `init_EPI`.

4.2.15 surfsmooth

Usage: `surfsmooth <func_prefix> <FWHM>`

> func_prefix – functional data prefix (eg.,smooth in func_smooth).
> FWHM – full-width half-maximum of the gaussian kernel convolved with the surface data.

This spatially-smooths cortical data along the surface mesh, estimated by Freesurfer.

Prerequisites: `init_EPI`, `vol2surf`.

4.2.16 surf2vol

Usage: `surf2vol <func_prefix> <target_prefix>`

> func_prefix – functional data prefix (eg.,smooth in func_smooth).
> target_prefix – target data prefix (eg.,smooth in func_smooth).

This projects surface data back into a functional volume with the same properties as <target_prefix>.

Prerequisites: `init_EPI`, `vol2surf`.

4.2.17 vol2surf

Usage: `vol2surf <func_prefix>`

> func_prefix – functional data prefix (eg.,smooth in func_smooth).

Projects functional data from volume space to a Freesurfer generated cortical mesh.

Prerequisites: `init_EPI`.

4.2.18 volsmooth

Usage: `volsmooth <func_prefix> <mask_prefix> <FWHM>`

> func_prefix – functional data prefix (eg., smooth in func_smooth).
> mask_prefix – mask data prefix (eg., EPI_mask in anat_EPI_mask).
> func_prefix – functional data prefix (eg.,smooth in func_smooth).

Re-samples a mask containing one or more labels to the functional data and smooths within unique values. All zero values in the mask are zeroed out in the output. The output of this can be combined with the outputs of `surfsmooth` & `surf2vol` using `combine_volumes`.

Prerequisites: `init_EPI`.

4.3 Quality Control

5 Further Reading