Preprocessing

August 27, 2014

1 Preprocessing

A few easy steps to preprocessing happiness (hopefully)...

- copy the contents of /scratch/tr332/[username] to your home directory
- add to your /.bashrc file the following to the *PATH* variable
  - /home/**username**/fmri_spt/:
  - app/fsl/fsl-4.1.7/bin:
  - home/**username**/abin/:
  - app/wbic/bin:
  - app/wbic/script:
  - app/vtkCISG/bin:
  - app/vtk/bin:
  - app/system/bin:
  - app/system/script:
  - app/spm/bin:
  - app/PETtools/bin:
  - app/PETtools/root/bin:
  - app0/generic/n1ge6_1/bin/lx24-x86:
  - app/misc/bin:
  - app/jvm1.1.7/TalairachDaemon/bin:
  - app/jdk1.5.0_03/bin:
  - app/java/bin:
  - app/intel/compiler60/ia32/bin:
  - app/idl_8.1/idl/bin/:
  - app/fsl/fsl-4.1.2/bin:
  - app/fsl/bin:
  - app0/i486-pc-linux/freesurfer5/freesurfer-5.0.0/bin:
  - app0/i486-pc-linux/freesurfer5/freesurfer-5.0.0/fsfast/bin:
  - app/fsl/fsl-4.1.2/bin:

- app0/i486-pc-linux/freesurfer5/freesurfer-5.0.0/mni/bin:
- app0/i486-pc-linux/freesurfer5/freesurfer-5.0.0/bin:
- app/ctn/bin:
- app/camres/bin:
- app/AnalyzeTools/bin:
- app/analyze/BIR/bin:
- app/AIR/bin:
- app/AIR:
- app/AIR/bin:
- app/afni:
- app/abi:
- home/tr332/bin/i486-pc-linux:
- bin:
- usr/bin:
- usr/sbin:
- usr/X11R6/bin:
- .:
- app/jvm1.1.7/BeanShell/scripts:
- app/matlab2008a/bin:
- app/matlab2007b/bin:
- app/matlab7/bin:
- app/matlab2006b/bin:
- app/matlab6/bin:
- app/mpitool:
- app/mricro:
- app/mricron:
- app/rpm:
- app/vnc/bin:

• for the *MATLABPATH* variable

- /home/tr332/fmri_spt/code_bin/:
- /app/abi:
- /app/abi/Plsgui:

• restart the terminal!

• for each scan reorientate the images (with fslreorient2std)

• use the newest SPM version for normalisation using and intermediate DARTEL-generated template

- run speedypp.py. This will correct for oblique acquisition and do skull stripping, slice timing correction, motion correction, regression out of motion parameters and CSF signal, high-pass filter, despiking and coregistration. I would suggest running this on the grid engine to save time using doSpeedypp.py.
- use the inverse warp generated from the normalisation step to convert to MNI space

2 Quality control

- Quality control.the resampling may chop off bits of the front of bottom of the brain. If this is the case, you will need to zeropad on the affected side and resample to this template. Look at afni's 3dZeropad to do this (eg 3dZeropad -I 20 –prefix template_zeropad_inf.nii template.nii). Then you warp to the original (unzeropadded template). Again, I attached modified script showing you an example.
- Excluding by movement. There is a script called fd.sh which extracts framewise displacement for a scan. Unfortunately it requires python packages not on the usual WBIC server. However, I have a virtual server set up called cluster4-0 that has the relevant packages. I will contact the WBIC to get you access to it, or alternatively you can let me know when you need this and I can run the script. It only takes a few seconds.

3 Parcellation

Parcellation is carried out using the script *tsExtractorScript.py*. This script masks the functional scan by both a structural and functional mask, then returns a series of timeseries using fsl commands. Following masking, the script checks the number of voxels in each parcel and proceeeds only if there are more than 10 voxels in the parcel. Otherwise, it returns a line of NA values instead of a timeseries. The final file is tagged with "_ts.txt" and contains ordered timeseries **in rows**.

Mask generation and timeseries extraction are wrapped in the script *runParcellate_withgrid.sh* which submits each subject's process individually to the grid engine. This script requires a warp file between the study specific template and MNI space, which can be generated using FNIRT. To save on scratch space, the *runParcellate_withgrid.sh* script copies each subjects' scans from a data directory to scratch. The paths will need adjusting in the *runParcellate_withgrid.sh* preamble.

4 Analysis of Motion

Ameera's pipeline includes a number of useful motion correction tools. These are all wrapped in *runMotionCheck.sh* for an individual, and *doCheckMovement.sh* which runs *runMotionCheck.sh* for all subjects by diagnostic group. For each subject the script produces cloud and loess plots, and correlation values for each individual. The motion correction scripts expect timeseries to be arranged in columns and use MATLAB which expects NA values to be coded 'NaN', so these changes are made at the beginning of *runMotionCheck.sh.* Part of the analysis is run on cluster4-0 (requiring up-to-date versions of numpy and python). Because of this, subjects are run in series rather than in parallel, however each subject only takes a minute or two.

All these files need transferring locally for generation of montaged images for quality control use (I tend to use Filezilla with a filename filter). The script *QCmontage.py* draws a sagittal view of a subject's functional network and creates a montage with the loess and cloud plots, and histograms of edge lengths and connection strengths. This script depends on the maybrain python package (http://code.google.com/p/maybrain/) to draw the network and histograms. There are some issues with multiple plotting of networks, so it is best to wrap subjects within bash rather than python using *doQCmontages.sh.*

In addition, it is possible to produce a Satterthwaite plot for a whole group using *satterthwaite.py.* This requires access to cluster4-0 to use the python scipy package.

5 Wavelet correlation

Wavelet correlation is based on Sophie Achard's wavebrainer package in R, run in python using rpy2 to link numpy and R. This is set up to use on cluster4-0. Rather than using the default brainwaver functions to write out association matrices, I have written the script to write NA values for blank timeseries.