**Belyk’s Abridged guide to SPM12 — GUI**

1. Organize data
   1. Make a well nested folder hierarchy e.g…
      1. MyProject
         1. Subject1
            1. anat
            2. bold1
            3. bold2…
         2. Subject2…
   2. Launch SPM
      1. Open MATLAB
      2. Change MATLAB directory to spm12 installation
      3. >>addpath D:\Michel\Phonotopy\spm12 (…or wherever)
      4. >>spm
      5. Click fMRI
   3. Convert images to 3D (not 4D) nifty (.nii) files
      1. Click DICOM importer (opens new window)
      2. Interact with top panel
      3. Items with <-X require user input
         1. Double click DICOM files
            1. Use left panel to browse folders
            2. User right panel to click & select .dcm files
            3. Done
         2. Double click Output Directory
            1. Use left panel to browse folders
            2. Use right panel to select where to place .nii files
            3. Done
         3. Run (green flag)
         4. Makes 1 .nii file per acquisition
         5. Consider MRCron if there is some problem

Note: Can work with 4D nifii. In GUI set “frames” dialogue to 1:no\_volumes

1. **Re-origin** anatomical scans so the anterior commissure is near 0,0,0
   1. Click Display->browse for anatomical
   2. Click Add overlay->browse for first volume of each functional run
   3. Check that these images start more or less coregistered
   4. Remove overlay
   5. Rotate anatomical to roughly the usual orientation by typing values into the pitch (x-rotation), roll (y-rotation), yaw (z-rotation) fields (units in radians because dumb).
   6. Move crosshair to the anterior commissure
   7. Note values in mm field
   8. Type these values in the fields below with the sign inverted
   9. Enter
   10. Type 0 0 0 in top mm field
   11. Enter
   12. Origin (check that crosshairs snaps to AC)
   13. Reorient image->select anat.nii, 1 and all volumes of all functionals
   14. Use Add Overlay to confirm that this has worked well
   15. I know this sucks… but take your time anyway. Having to do it all over again sucks harder.
   16. Repeat for each subject

**Pre-processing**

1. **Realignment** to correct for head motion within functional runs
   1. This estimates the rigid body transformation to align each volume within a run. Saves transform values to header without applying them.
   2. *Click realign: estimate* from dropdown menu
   3. Inputs
      1. Double click Data
      2. Double click session browse for and select bold.nii files
      3. Repeat X number of runs to process
      4. Check settings in upper window
      5. Run batch (green flag)
   4. Outputs
      1. list of x,y,z translations and rotations aligning each volume to the first in the series. (r\*.txt).
      2. Realignment parameters are also stored in image headers and will be implemented at the next call to *reslice*
   5. How to view and check this? Write some matlab plotter?
2. **Coregister** functional runs onto anatomical run (rigid-body transformation)
   1. Registers first volume of a functional run onto T1 anatomical, then applies the same transformation to subsequent volumes.
   2. Click *Coregister: estimate & reslice* from dropdown menu
   3. Inputs
      1. Reference Image <- T1 anatomical image
      2. Source image <- first functional image volume
      3. Other images <- the rest of the epi image volumes
   4. Outputs
      1. new images (prefixed *r*) implementing realignment and coregistration to folder containing source images.
   5. Move new images to a new folder
3. Spatial Smoothing
   1. Wasn’t necessary for the project that I tested this on. Will add later.
4. **Segment** gray matter, white matter and CSF from T1
   1. May be useful for masking or VBM later, but more importantly makes \*\_seg\_sn.mat file as a that is useful for normalization to MNI template.
   2. Click  *Segment*
   3. Inputs
      1. Data<-T1 anatomical
      2. Deformation fields<- forward + inverse
         1. Forward needed to normalize T1
         2. Optional: Inverse needed to normalize surface maps
   4. Outputs
      1. c1\*.nii (gray matter)
      2. c2\*.nii (white matter)
      3. c3\*.nii (CSF)
      4. c4\*.nii (pia/dura matter)
      5. c5.nii (volume mask)
      6. \*\_seg8.mat (normalization transform data)
      7. y\_\*\_sn.nii (forward deformation field)
      8. iy\_\*\_sn.nii (inverse deformation field)
5. **Normalize** scans to MNI template.
   1. Applies transform calculated in segmentation step to T1 anatomical and functional runs that are already registered to it.
   2. Click *Normalise: Write* from dropdown menu
   3. Input
      1. Deformation field<- y\_\*\_sn.nii produced by segmentation
      2. Images to write<-coregistered functional images (r\*.nii)
         1. Select all images for one participant
         2. Including anatomical
   4. Output
      1. MNI normalized functional images (w\*.nii)
   5. Move these to folder *mni*
6. **Manually check registration**
   1. Check Reg ->browse for images

Or…

* 1. Display->browse for wanat
  2. Add Overlay->browse for wbolds

**First-Level Analysis**

1. **Fixed effects analysis** for individual subject maps
   1. Runs separate general linear models for each subject. Parameter estimates from this analysis are inputs to the group level random effects analysis.
   2. Click *Specify 1-st level*
   3. this is going to be kludgy, will look for better alterantives. Might be better if some of this is already specificed in a .txt or .mat file for command line importing.
      1. Directory<-participant directory to save design matrix
      2. Units of design<- Scans (measure in volumes, TRs, whatever)
      3. Interscan interval<- TR (in seconds)
      4. Data & Design:  *New Subject/Session*
         1. Add new subject/session, one per run
         2. Scans<-browse for normalized data (mni\w\*.nii)
         3. Conditions<-New Condition X # conditions/run
         4. Name<- Thing what it’s called
         5. Onsets<- The time(s) that each block of condition starts
         6. Durations<-How long each block is
      5. Regressors (add motion correction parameters as covariates)
         1. Multiple regressors<-browse for coreg/rp\*.txt
      6. File<-Save Batch (to facilitate re-checking/re-running later)
      7. View summary of design matrix
         1. Main window<-Review<-browse for SPM.mat
   4. Click Estimate<- browse for SPM.mat
   5. Outputs one beta\_000*k*.nii for each of *k* regressors specified above. The last seven are motion covariates and the intercept, which are of little interest
   6. If multiple runs are estimated at once, this can lead to clumsy file names. With X conditions of interests
      1. beta0001 through beta000X = conditions of interest in run 1
      2. beta000[X+1] through beta000[X+6] = motion correction
      3. beta000[X+7] through beta000[2X+7] = conditions in run 2
      4. etc… This is bullshit.
      5. The final beta\*.nii files are intercepts and of little interest.
2. **Define Contrasts**
   1. Click *Results->*Browse for SPM.mat saved in previous step
   2. Define new contrast
      1. Give a sensible name
      2. Provide contrast weights, examples below
         1. **1, -1,0,0** to contrast the first two of four conditions
         2. **-1,-1,1,1** to contrast the first two against last two
         3. Note: weight motion parameters and intercepts as 0
      3. Submit
      4. Done
      5. Answer questions about what thresholds you want to apply.

*CMD-LINE NOTES*

SPM uses a job manager which is basically a way to execute commands who’s details and settings are saved in a .mat file. This should somehow make batch scripting easier, but how to interact with this is still a little unclear to me.

Complete each step once for a sample participant

Save each step as a batch

Jobman(jobs,”run”) #this executes the batch

jobs is a matlab struct object that can be manipulated

chang it’s filepaths to point to new data then execute

do this in a loop

<http://andysbrainblog.blogspot.ca/2012/11/spm-jobman.html>

My approach to SPM scripting (see phonotopy\_preproc.m as sample) is to export each step from GUI as a job. I’ve already loaded these into MATLAB and given them more sensible names. These can now be loaded, manipulated to point to the right data, then executed. Set up files and directories with straightforward names that are easy to loop through (as described above in section 0).