neuroimaging core Documentation

Release 0.1.1

Dianne Patterson

Apr 07, 2020

Table of Contents

Maintainer: Dianne Patterson dkp @ email.arizona.edu

Date Created: 2018_06_14

Date Updated: 2019_11_14

These pages provide documentation for the neuroimaging core at the University of Arizona. The focus is on approaches used and/or developed at the University of Arizona.

At the top of each page is a record of the maintainer, date created and date updated to help the user identify information that is likely to be stale, and notify the maintainer.

Pages that describe processing using particular neuroimaging software should include software version and OS version information.

Want PDFS? Read the Docs has the ability to output documentation in other formats: pdf and epub. On the bottom left it says `Read the Docs v:latest`. Under `Downloads`, choose one of the `latest` options (e.g. PDF). The PDFs have much nicer formatting than if you simply "save as" pdf in your browser. The only downside is that the pdf includes everything on the website, so you'll want to use Preview or Adobe Acrobat or something similar to extract the portion of the documentation you want. Of course, you could just print the range of pages you want as hard copies.

Atlases

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2019_10_31
Date Updated: 2020_02_16
Tags: Neuroimaging software, Atlases, FSL
OS: Mostly UNIX (e.g., Mac or Linux)

- Links to the atlas resources can be found on the bookmarks page *Brain Atlases*.

- FSL comes with a range of atlases prepared for viewing in FSLeyes. Learn more on the Atlases page.

- FSL atlases can be created by following these atlas reference rules

1.1 How to Download and Install Additional FSL Atlases

Below I provide several additional atlases to display in FSLeyes. To use these atlases, download the zip file by clicking on the atlas title. Place the unzipped folder and xml file(s) in the FSL atlases directory (e.g., /usr/local/fsl/data/atlases). The next time you start fsleyes, the atlas should be available in the atlas panel.

1.2 BIP

The BIP language atlas is based on: Patterson, D. K., Van Petten, C., Beeson, P., Rapcsak, S. Z., & Plante, E. (2014). Bidirectional iterative parcellation of diffusion weighted imaging data: separating cortical regions connected by the arcuate fasciculus and extreme capsule. NeuroImage, 102 Pt 2, 704–716. http://doi.org/10.1016/j.neuroimage.2014.08.032

The regions and tracts are described in the Readme.txt

The software for running BIP is available as a containerized BIDS app. See *bip container* for a description and the bipbids bitbucket site for the code. Note that the Singularity recipe contains code to use NVIDIA GPUs on out HPC. The Docker container, which can be pulled from dockerhub `docker pull diannepat/bip` does not contain the GPU code.

1.3 HCP-MMP1 and HCP-MMP1_cortices

The HCP-MMP1 atlas is based on the asymmetrical version of the MNI projection of the HCP-MMP1 (MMP_in_MNI_corr.nii.gz) available on figshare and neurovault.

In each hemisphere, Glasser divides 180 "areas" into 22 separate "regions". Here I refer to the 180 areas as **regions** to be consistent with other atlases where "region" generally refers to the smallest partition of interest. I call the 22 larger partitions **cortices**.

TIn this HCP-MMP1 atlas the 180 regions are numbered 1-180. On the right the regions are numbered 201-380 so that 201 is the right homologue of 1; 262 is the right homologue of 62, etc.) Note that MRtrix3 renumbers the values on the right to go from 181 to 360 to avoid the discontinuity (i.e., unused values between 181 and 199).

Each of the 180 regions occupies one of 22 cortices which are displayed in a separate atlas: HCP-MMP1_cortices. These are numbered 1-22 on the left and 101-122 on the right, in keeping with the original strategy.

Detailed information about the regions and cortices are available in the Glasser 2016 Supplemental file: "Supplementary Neuroanatomical Results For A Multi-modal Parcellation of Human Cerebral Cortex". I have made the the following helpful files available:

1. The final table of 180 regions from that supplemental file available as an Excel sheet: Glasser_2016_Table.xlsx

2. HCP-MMP1_UniqueRegionList.csv providing information from the final table in addition to a center of gravity in voxel coordinates and a volume in cubic mm for each of the 360 regions (180 in each hemisphere).

3. A text file naming the 22 cortices and how they are grouped as per descriptions in the supplemental material.

4. HCP-MMP1_cortex_UniqueRegionList.csv providing the center of gravity in voxel coordinates and the volume in cubic mm for each of the 44 cortices (22 in each hemisphere).

This is the accompanying paper: Glasser, M. F., Coalson, T. S., Robinson, E. C., Hacker, C. D., Harwell, J., Yacoub, E., et al. (2016). A multi-modal parcellation of human cerebral cortex. Nature, 1–11. http://doi.org/10.1038/nature18933

1.4 GreciusFunc

This is a 2mm probabalistic atlas with 13 functional regions defined. It includes both cortex and cerebellum. The data for each roi were downloaded from http://findlab.stanford.edu/functional_ROIs.html

This is the accompanying paper:

Shirer, W. R., Ryali, S., Rykhlevskaia, E., Menon, V., & Greicius, M. D. (2012). Decoding Subject-Driven Cognitive States with Whole-Brain Connectivity Patterns. Cerebral Cortex. http://doi.org/10.1093/cercor/bhr099

1.5 YeoBuckner7 and YeoBuckner17

These are 2mm probabalistic atlases converted from Freesurfer space to FSL standard space with the cortex and cerebellum combined. The data comes from http://surfer.nmr.mgh.harvard.edu/fswiki/CorticalParcellation_Yeo2011 and http://www.freesurfer.net/fswiki/CerebellumParcellation_Buckner2011 The two atlases are:

1. YeoBuckner7: The 7 functional networks

2. YeoBuckner17: Finer divisions of the 7 networks into a total of 17 networks (which are not always proper subsets of the first 7)

These are the accompanying papers:

Yeo, B. T. T., Krienen, F. M., Sepulcre, J., Sabuncu, M. R., Lashkari, D., Hollinshead, M., et al. (2011). The organization of the human cerebral cortex estimated by intrinsic functional connectivity. Journal of Neurophysiology, 106(3), 1125–1165. http://doi.org/10.1152/jn.00338.2011

Buckner, R. L., Krienen, F. M., Castellanos, A., Diaz, J. C., & Yeo, B. T. T. (2011). The organization of the human cerebellum estimated by intrinsic functional connectivity. Journal of Neurophysiology, 106(5), 2322–2345. http://doi.org/10.1152/jn.00339.2011

BIDS

Author/Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu

Date Created: 2018_12_07

Date Updated: 2019_09_14

Tags: BIDS, standards, containers

OS: UNIX (e.g., Mac or Linux)

2.1 BIDS Containers

Author/Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu

Date Created: 2019_07_30

Date Updated: 2020_04_06

Tags: BIDS, containers

OS: UNIX (e.g., Mac or Linux)

2.1.1 HeuDiConv

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu

Date Created: 2018_12_07

Date Updated: 2020_02_27

Tags: BIDS, standards, containers, conversion, DICOM, NIFTI, python3

OS: UNIX (e.g., Mac or Linux), verified with heudiconv version 0.6.0.dev1 and earlier

Acknowledgements: Tom Hicks, Adam Raikes, Aneta Kielar, and the heudiconv and reproin teams

Introduction

HeuDiConv (Heuristic Dicom Conversion) provides sophisticated and flexible creation of BIDS datasets. It calls *dcm2niix* to convert the DICOMS to NIFTI with sidecar JSON files. It also produces the additional files that BIDS expects (e.g., CHANGES, dataset_description.json, participants.tsv, README etc.). Similar tools include:

- BIDSkit

- DAC2BIDS

- Dcm2Bids

- dcm2niix batch. See also this excellent example of working through the whole process.

HeudiDiConv runs in a container (Docker or Singularity) or by pip install (e.g., an Anaconda environment). Instructions for pip installs are available in the official readme.

- On this page, I provide tutorials for two of the heuristic files provided with HeuDiConv.

    - The first heuristic is called *convertall.py*.

    - The second is called *reproin.py*.

- **convertall.py** is better if you already have DICOM files to convert. However, *convertall.py* requires that you modify a Python module to specify your particular naming scheme to HeuDiConv. This process is detailed below with explicit examples for typical U of A sequences. It isn't horrible. Don't assume that the *Region* and *Exam* assigned at the scanner will be available to convertall.py in the same way they are available to reproin.py.

- **reproin.py** takes a different approach. *reproin* requires you to follow a particular naming scheme at the scanner instead of modifying a Python module. If you have not yet named your scanner sequences, then it is worth using the reproin naming scheme. Overall, reproin is easier, but is more limited in its support for image types that are not yet supported by BIDS.

    - **reproin.py** is "part of the ReproNim Center suite of tools and frameworks. Its goal is to provide a turnkey flexible setup for automatic generation of shareable, version-controlled BIDS datasets from MR scanners." reproin site

Links

A number of useful resources are available for you to consult. Here are the ones I've found.

- HeuDiConv github site

- Stanford Center for Reproducible Neuroscience Tutorials.

    - Part 1A Manual Conversion of Dicoms,

    - Part 1B: Automated Conversion of Dicoms,

    - Part 2A: HeuDiConv Conversion of Dicoms.

- convertall.py

    - James Kent video walkthrough

    - James Kent walkthrough slides: Docker and singularity examples

    - Nipype Workshop, March 2017 slides

- reproin.py

    - main reproin site

    - reproin heuristic file

- reproin cheetsheet
- A test dataset to use with the convertall.py tutorial on this page dicoms_heudiconv.zip
- A test dataset to use with the reproin tutorial on this page dicoms_reproin_UA.zip
- A final version of convertall.py
- An example script for running the iterative part of a convertall.py script: hdc_wrap.sh. Read over the script! It makes some hardwired assumptions about the way your dicoms are named and nested (especially in the case of using a session).

> **Warning:** Be careful to name sessions using only dashes and not underscores. That is `itbs-pre` will work fine, but `itbs_pre` will cause you pain in later processing.

---

convertall.py

Run HeuDiConv in a 3-step process:

- *Step1* By passing some path information and flags to HeuDiConv, we generate a heuristic (translation) file, *convertall.py* and some associated descriptor text files. These all get placed in a hidden directory, **.heudiconv** under the output directory.

- *Step2* We create a subdirectory under the output directory and call it **code**. Here we place a copy of the Python3 heuristic/translation file (in this case, *convertall.py*) which we will modify by hand to specify the output names and directories, and the input DICOM characteristics. A file called *dicominfo.tsv* contains a listing of the DICOM characteristics that we can use as criteria. Most of the description in this section pertains to editing *convertall.py* for your own dataset.

- *Step3* Having revised *convertall.py*, we now call HeuDiConv to run it on one or more subjects and sessions. Each time we run it, an additional subdirectory is created under *.heudiconv* that records the details of that conversion. We can run it again on new subjects and sessions as they become available. *convertall.py*, along with the .heudiconv hidden directory, provide provenance information that you should retain with your dataset. You can rename your heuristic file, which may be useful if you have multiple heuristic files for the same dataset.

The suggested project structure looks like this:

- In a top-level project directory (e.g., MyProject), you keep a Dicom directory and a Nifti directory.

- The Nifti directory contains the following:

  - The hidden *.heudiconv* directory which records provenance information about the DICOM to NIFTI conversion for each subject or subject session.

  - The *code* directory (which the *bids-validator* ignores) and where you should put your manually modified convertall.py

  - All the subject directories for the unprocessed NIFTI files (The subject directories are suitable for export to data repositories.)

  - A *derivatives* directory to hold the output of later processing with other apps:

```
MyProject
|-- Dicom
|    `-- 219
|        |-- ASL_3D_tra_iso_22
|        |-- AX_DE_TSE_14
|        |-- Bzero_verify_PA_17
|        |-- DTI_30_DIRs_AP_15
```

(continues on next page)

```
|       |-- DTI_30_DIRs_AP_TENSOR_16
|       |-- JJWANGS_pCASL_PLD1800_24
|       |-- JJWANGS_pCASL_PLD1800_34
|       |-- Localizers_1
|       |-- MoCoSeries_19
|       |-- MoCoSeries_31
|       |-- Perfusion_Weighted_23
|       |-- Perfusion_Weighted_25
|       |-- Perfusion_Weighted_33
|       |-- Perfusion_Weighted_35
|       |-- Post_TMS_ASL_3D_tra_iso_32
|       |-- Post_TMS_restingstate_30
|       |-- T1_mprage_1mm_13
|       |-- field_mapping_20
|       |-- field_mapping_21
|       `-- restingstate_18
|-- Nifti
|   |-- .bidsignore
|   |-- .heudiconv
|   |   `-- 219
|   |-- CHANGES
|   |-- README
|   |-- code
|   |   |-- __pycache__
|   |   `-- convertall.py
|   |-- dataset_description.json
|   |-- participants.tsv
|   |-- sub-219
|   |   `-- ses-itbs
|   `-- task-rest_bold.json
|   |--derivatives
`-- dicominfo.tsv
```

- HeuDiConv does not need the *derivatives* directory. It is included here so you can see how the whole structure will be laid out. You can name your *project*, and the *Nifti* directory as you wish. The BIDS standard expects you to put your output under a directory named *derivatives*. If you wish to hold the results of different processing pipelines, simply create subdirectories under *derivatives* for each pipeline (fmriprep will assume it should put its output in a subdirectory under derivatives, so you don't have to provide it with this nesting). You may have to specify this nesting for other containers.

- You are likely to have `IMA` files if you copied your images from the scanner to an external drive, but you are likely to have `.dcm` files if you exported your images from Horos or Osirix. Watch out for capitalization differences in the sequence names (`.dcm` files are typically lower case, but `IMA` files are typically upper case).

- Heudiconv will extract age and sex info from the DICOM header. If there is any reason to believe this information is wrong in the DICOM header (for example, it was made-up because no one knew how old the subject was, or it was considered a privacy concern), then you need to check it. If you have Horos (or another DICOM editor), you can check and edit the values there so they will be correct when you export them.

- If you have multiple sessions at the scanner, you should create an *Exam* folder for each session. This will help you to keep the data organized and *Exam* will be reported in the *study_description* in your dicominfo.tsv, so that you can use it as a criterion.

> **Warning:** Do not assume that different subjects have the same values in the DICOM headers. That is, if you develop a convertall.py on one subject, try it and carefully evaluate the results on your other subjects. This is especially true if you already collected the data before you started thinking about automating the output. Every

time you run HeuDiConv with convertall.py, a new dicominfo.tsv file is generated. Inspect this for differences in protocol names and series descriptions etc.

Warning: I recently (03/11/2019) found that heudiconv failed if the data I exported from Horos was not decompressed. This was especially confusing because dcm2niix succeeded on this data... hmm.

Warning: If you have combined multiple sessions in one dicom folder, heudiconv will fail to run and will complain about "conflicting study identifiers". You can get around the problem by figuring out which dicoms are from different studies and separating them so you deal with one set at a time.

The `.heudiconv` hidden directory

- **The Good** Every time you run conversion to create the nifti files and directories, a record of what you did is recorded in the *.heudiconv* directory. This includes a copy of the *convertall.py* module that you ran for each subject and session.

- **The Bad** If you rerun *convertall.py* for some subject and session that has already been run, heudiconv quietly uses the conversion routines it stored in *.heudiconv*. This can be really annoying if you are troubleshooting *convertall.py*.

- **More Good** You can remove subject and session information from *.heudiconv* and run it fresh. In fact, you can entirely remove the *.heudiconv* directory and still run the *convertall.py* you put in the *code* directory. This will give you a fresh start. It obviously also means you can send someone else the convertall.py for a particular project and they can run it too.

Walkthrough of HeuDiConv Run

- The following examples assume you are at the bash shell prompt. You need to download dicoms_heudiconv.zip and unzip it into a directory. Let's call that directory *MyProject* (see above).

- The data in *dicoms_heudiconv.zip* was exported from Horos (so it uses lower case and the .dcm extension)

- Here we assume **you have downloaded Docker and have it running**. Then you need to pull the HeuDiConv Docker container to your machine:

```
docker pull nipy/heudiconv
```

Step 1

From the *MyProject* directory, run the following Docker command to process the dcm files that you downloaded and unzipped for this tutorial. The subject number is 219:

```
docker run --rm -it -v ${PWD}:/base nipy/heudiconv:latest -d /base/Dicom/{subject}/*/
↪*.dcm -o /base/Nifti/ -f convertall -s 219 -c none
```

> **Warning:** The above docker command works in bash, but may not work in other shells. For example, zsh is uset by the form `{subject}` but bash actually doesn't mind.

- `--rm` means Docker should cleanup after itself (it removes the instance after it runs)

- `-it` means Docker should run interactively

- `-v ${PWD}:/base` binds your current directory to `/base` inside the container. You could also provide an **absolute path** to the *MyProject* directory.

- `nipy/heudiconv:latest` says to use the latest version of heudiconv.

- `-d /base/Dicom/{subject}/*/*.dcm` says to look for a directory called *Dicom* in the current directory (*MyProject*). Under *Dicom* HeuDiConv should look for the subject directory (e.g., 219). Actual dcm files will then be under any subdirectories at this level:

```
    Dicom
|-- 219
|   |-- ASL_3D_tra_iso_22
|   |-- AX_DE_TSE_14
|   |-- Bzero_verify_PA_17
|   |-- DTI_30_DIRs_AP_15
|   |-- DTI_30_DIRs_AP_TENSOR_16
|   |-- JJWANGS_pCASL_PLD1800_24
|   |-- JJWANGS_pCASL_PLD1800_34
|   |-- Localizers_1
|   |-- MoCoSeries_19
|   |-- MoCoSeries_31
|   |-- Perfusion_Weighted_23
|   |-- Perfusion_Weighted_25
|   |-- Perfusion_Weighted_33
|   |-- Perfusion_Weighted_35
|   |-- Post_TMS_ASL_3D_tra_iso_32
|   |-- Post_TMS_restingstate_30
|   |-- T1_mprage_1mm_13
|   |-- field_mapping_20
|   |-- field_mapping_21
|   `-- restingstate_18
```

- `-o /base/Nifti/` says to put the output in a directory called *Nifti* (i.e., MyProject/Nifti). If the *Nifti* directory does not exist, it will be created.

- `-f convertall` This creates a convertall.py template from an existing heuristic module. There are [other heuristic modules](#) , e.g., banda-bids.py, bids_with_ses.py, cmrr_heuristic.py, example.py, multires_7Tbold.py, reproin.py, studyforrest_phase2.py, test_reproin.py, uc_bids.py. convertall.py is a good default though.

- `-s 219` says the subject number is *219*. `219` will replace {subject} in the `-d` argument when Docker actually runs.

- `-c none` says we are not actually doing any conversion right now. Instead we are going to generate *dicominfo.tsv* and *convertall.py* in the hidden directory *.heudiconv* under your output directory, e.g. *MyProject/Nifti/.heudiconv*

- You should create a directory called *code* under your output directory (e.g., the *Nifti* directory) and copy *convertall.py* from *.heudiconv/info* to *code*. This is useful because you can look back at the original template in the hidden directory if you need to, but you can use *convertall.py* in the *code* directory to test your modifications.

- Step 1 only needs to be completed once correctly for each project.

Step 2

Now we want to modify three sections in *convertall.py*. You can download the final version here: convertall.py

The Modifications Explained

- Your goal is to produce a working *convertall.py* that will arrange the output in a BIDS directory structure. However, keep in mind that the hidden *.heudiconv* directory gets updated every time you run heudiconv. In addition, both your *code* directory and your *.heudiconv* directory provide valuable provenance information that should remain with your data. Once you create a working *convertall.py*, you can run it for different subjects and sessions (keep reading).

- The *convertall.py* template contains a lot of explanatory text for you to read. I have removed this from the final version to keep it short.

- I provide three section labels (1, 1b and 2) to facilitate exposition here. Each of these sections should be manually modified by you for your project.

  - **Section 1**

    * This *convertall.py* does not import all sequences in the example *Dicom* directory. This is a feature of heudiconv: You do not need to import scouts, motion corrected images or other dicoms of no interest.

    * You may wish to add, modify or remove keys from this section for your own data:

    ```python
    # Section 1: These key definitions should be revised by the user
    ################################################################
    # For each sequence, define a key and template using the create_key
    →function:
    # key = create_key(output_directory_path_and_name).
    #
    # The "data" key creates sequential numbers which can be for naming
    →sequences.
    # This is especially valuable if you run the same sequence multiple times
    →at the scanner.
    data = create_key('run-{item:03d}')
    t1w = create_key('sub-{subject}/{session}/anat/sub-{subject}_{session}_T1w
    →')
    tse = create_key('sub-{subject}/{session}/anat/sub-{subject}_{session}_
    →acq-tse_T2w')
    dwi = create_key('sub-{subject}/{session}/dwi/sub-{subject}_{session}_acq-
    →AP_dwi')
    fmap_rev_phase =  create_key('sub-{subject}/{session}/fmap/sub-{subject}_
    →{session}_dir-PA_epi')
    fmap_mag =  create_key('sub-{subject}/{session}/fmap/sub-{subject}_
    →{session}_magnitude')
    fmap_phase = create_key('sub-{subject}/{session}/fmap/sub-{subject}_
    →{session}_phasediff')
    func_rest = create_key('sub-{subject}/{session}/func/sub-{subject}_
    →{session}_task-rest_run-01_bold')
    func_rest_post = create_key('sub-{subject}/{session}/func/sub-{subject}_
    →{session}_task-rest_run-02_bold')
    asl = create_key('sub-{subject}/{session}/func/sub-{subject}_{session}_
    →acq-asl_run-01')
    asl_post = create_key('sub-{subject}/{session}/func/sub-{subject}_
    →{session}_acq-asl_run-02')
    ```

    * **Key**

      · Define a short key for each image sequence you wish to export. Note that you can use any key

names you want (e.g. *foo* would work as well as *fmap_phase*), but you need to be consistent, and you should choose key names that are short and easy to understand.

· The `key` name is to the left of the = for each row in the above example.

* **Template**

  · Use the variable `{subject}` to make the code general purpose, so you can apply it to different subjects in Step 3.

  · Use the variable `{session}` to make the code general purpose only if you have multiple sessions for each subject.

  · Once you use the variable `{session}`, ensure that a session gets added to the **output path**, e.g., `sub-{subject}/{session}/anat/` and the **output filename**: `sub-{subject}_{session}_T1w` for every image in the session. Otherwise you will get *bids-validator errors*.

  · It is up to you to define the output directories and file names according to the BIDS specification.

  · Note the output names for the fieldmap images (e.g., sub-219_ses-itbs_dir-PA_epi.nii.gz, sub-219_ses-itbs_magnitude1.nii.gz, sub-219_ses-itbs_magnitude2.nii.gz, sub-219_ses-itbs_phasediff.nii.gz), and that the reverse_phase encode dwi image (e.g., sub-219_ses-itbs_dir-PA_epi.nii.gz) is grouped with the fieldmaps. These naming strategies were less than obvious to me, but they are what the bids specification expects.

  · ASL and TSE images were not described in the BIDS specification as of this writing. Data that is not yet defined in the specification will cause the bids-validator to produce an error unless you include it in a *.bidsignore* file. The output filenames and organization used here for ASL and TSE images were created after consultation with the bids google group, but they still needed to be in the *.bidsignore* file.

* **data**

  · a key definition that creates sequential numbering

  · `03d` means *create three slots for digits* `3d`, *and pad with zeros* `0`.

  · This is useful if you have a scanner sequence with a single name but you run it repeatedly and need to generate separate files for each run. For example, you might define a single functional sequence at the scanner and then run it several times instead of creating separate names for each run.

Note: N.B. It is usually better to name your sequences explicitly rather than depending on sequential numbering. There will be less confusion later.

  · If you have a sequence with the same name that you run repeatedly WITHOUT the sequential numbering, HeuDiConv will overwrite earlier sequences with later ones.

  · To ensure that a sequence includes sequential numbering, you also need to add `run-{item:03d}` (for example) to the key-value specification for that sequence.

  · Here I illustrate with the t1w key-value pair:

  · If you started with:

  · `t1w = create_key('sub-{subject}/anat/sub-{subject}_T1w'),`

  · You could add sequence numbering like this:

  · `t1w = create_key('sub-{subject}/anat/sub-{subject}_run-{item:03d}_T1w').`

· Now if you run several T1w images using the exact same protocol, each will get a separate run number like this:

· *sub-219_ses_run-001_T1w.nii.gz, sub-219_ses_run-002_T1w.nii.gz* etc.

– **Section 1b**

* Based on your chosen keys, create a data dictionary called *info*:

```
# section 1b: This data dictionary (below) should be revised by the user.
##########################################################################
# Enter a key in the dictionary for each key you created above in section
↪1.
info = {data: [], t1w: [], tse: [], dwi: [], fmap_rev_phase: [], fmap_
↪mag: [], fmap_phase: [], func_rest: [], func_rest_post: [], asl: [],
↪asl_post: []}
# The following line does no harm, but it is not part of the dictionary.
last_run = len(seqinfo)
```

* Enter each key in the dictionary in this format `key:    []`, for example, `t1w:    []`.

* Separate the entries with commas as illustrated above.

– **Section 2**

* Define criteria for identifying each DICOM series that corresponds to one of the keys you want to export:

```
# Section 2: These criteria should be revised by user.
##########################################################
# Define test criteria to check that each DICOM sequence is correct
# seqinfo (s) refers to information in dicominfo.tsv. Consult that file
↪for
# available criteria.
# Here we use two types of criteria:
# 1) An equivalent field "==" (e.g., good for checking dimensions)
# 2) A field that includes a string (e.g., 'mprage' in s.protocol_name)

for idx, s in enumerate(seqinfo):
    if ('mprage' in s.protocol_name) and (s.dim3 == 176):
        info[t1w].append(s.series_id)
    if ('TSE' in s.protocol_name):
        info[tse].append(s.series_id)
    if ('DTI' in s.protocol_name) and (s.dim3 == 74) and (s.dim4 == 32):
        info[dwi].append(s.series_id)
    if ('verify_P-A' in s.protocol_name):
        info[fmap_rev_phase] = [s.series_id]
    if ('field_mapping' in s.protocol_name) and (s.dim3 == 64):
        info[fmap_mag] = [s.series_id]
    if ('field_mapping' in s.protocol_name) and (s.dim3 == 32):
        info[fmap_phase] = [s.series_id]
    if ('restingstate' == s.protocol_name):
        info[func_rest].append(s.series_id)
    if ('Post_TMS_restingstate' == s.protocol_name):
        info[func_rest_post].append(s.series_id)
    if ('ASL_3D_tra_iso' == s.protocol_name):
        info[asl].append(s.series_id)
    if ('Post_TMS_ASL_3D_tra_iso' == s.protocol_name):
        info[asl_post].append(s.series_id)
return info
```

* To define the criteria, look at *dicominfo.tsv* in *.heudiconv/info*. This file contains tab-separated values so you can easily view it in Excel or any similar spreadsheet program. *dicominfo.tsv* is not used programmatically to run heudiconv (i.e., you could delete it with no adverse consequences), but it is very useful for defining the test criteria for section 2 of *convertall.py*.

* Some values in *dicominfo.tsv* might be wrong. For example, my reverse phase encode sequence with two acquisitions of 74 slices each is reported as one acquisition with 148 slices (2018_12_11). Hopefully they'll fix this. Despite the error in *dicominfo.tsv*, dcm2niix reconstructs the images correctly.

* You will be adding, removing or altering values in conditional statements based on the information you find in *dicominfo.tsv*.

* `seqinfo` (s) refers to the same information you can view in *dicominfo.tsv* (although seqinfo does not rely on *dicominfo.tsv*).

* Here we illustrate two types of criteria:

    · `s.dim3 == 176` is an **equivalence** (e.g., good for checking dimensions for a numerical data type). For our sample T1w image to be exported from DICOM, it must have 176 slices in the third dimension.

    · `'mprage' in s.protocol_name` says the protocol name string must **include** the word *mprage* for the *T1w* image to be exported from DICOM. This criterion string is case-sensitive.

* `info[t1w].append(s.series_id)` Given that the criteria are satisfied, the series should be named and organized as described in *Section 1* and referenced by the info dictionary. The information about the processing steps is saved in the *.heudiconv* subdirectory.

* Here I have organized each conditional statement so that the sequence protocol name comes first followed by other criteria if relevant. This is not necessary, though it does make the resulting code easier to read.

Step 3

* You have now done all the hard work for your project. When you want to add a subject or session to *MyProject*, you only need to run this third step for that subject or session (A record of each run is kept in .heudiconv for you):

```
docker run --rm -it -v ${PWD}:/base nipy/heudiconv:latest -d /base/Dicom/{subject}
↪/*/*.dcm -o /base/Nifti/ -f /base/Nifti/code/convertall.py -s 219 -ss itbs -c␣
↪dcm2niix -b --minmeta --overwrite
```

* The first time you run this step, several important text files are generated (e.g., CHANGES, dataset_description.json, participants.tsv, README etc.). On subsequent runs information may be added (e.g., *participants.tsv* will be updated). Other files, like the *README* and *dataset_description.json* should be filled in manually after they are first generated.

* This Docker command is slightly different from the previous Docker command you ran.

    – `-f /base/Nifti/code/convertall.py` now tells HeuDiConv to use your revised *convertall.py* in the *code* directory.

    – In this case, we specify the subject we wish to process `-s 219` and the name of the session `-ss itbs`.

        * We could specify multiple subjects like this: `-s 219 220 -ss itbs`

    – `-c dcm2niix -b` indicates that we want to use the dcm2niix converter with the -b flag (which creates BIDS).

- `--minmeta` ensures that only the minimum necessary amount of data gets added to the JSON file when created. On the off chance that there is a LOT of meta-information in the DICOM header, the JSON file will not get swamped by it. fmriprep and mriqc are very sensitive to this information overload.

- `--overwrite` This is a peculiar option. Without it, I have found the second run of a sequence does not get generated. But with it, everything gets written again (even if it already exists). I don't know if this is my problem or the tool... but for now, I'm using `--overwrite`.

Step 3 should produce a tree like this:

```
|-- dataset_description.json
|-- participants.tsv
|-- sub-219
|   `-- ses-itbs
|       |-- anat
|       |   |-- sub-219_ses-itbs_T1w.json
|       |   |-- sub-219_ses-itbs_acq-tse_T2w1.json
|       |   |-- sub-219_ses-itbs_acq-tse_T2w1.nii.gz
|       |   |-- sub-219_ses-itbs_acq-tse_T2w2.json
|       |   |-- sub-219_ses-itbs_acq-tse_T2w2.nii.gz
|       |   `-- sub-219_ses_itbs_T1w.nii.gz
|       |-- dwi
|       |   |-- sub-219_ses-itbs_acq-AP_dwi.bval
|       |   |-- sub-219_ses-itbs_acq-AP_dwi.bvec
|       |   |-- sub-219_ses-itbs_acq-AP_dwi.json
|       |   `-- sub-219_ses-itbs_acq-AP_dwi.nii.gz
|       |-- fmap
|       |   |-- sub-219_ses-itbs_dir-PA_epi.json
|       |   |-- sub-219_ses-itbs_dir-PA_epi.nii.gz
|       |   |-- sub-219_ses-itbs_magnitude1.json
|       |   |-- sub-219_ses-itbs_magnitude1.nii.gz
|       |   |-- sub-219_ses-itbs_magnitude2.json
|       |   |-- sub-219_ses-itbs_magnitude2.nii.gz
|       |   |-- sub-219_ses-itbs_phasediff.json
|       |   `-- sub-219_ses-itbs_phasediff.nii.gz
|       |-- func
|       |   |-- sub-219_ses-itbs_acq-asl_run-01.json
|       |   |-- sub-219_ses-itbs_acq-asl_run-01.nii.gz
|       |   |-- sub-219_ses-itbs_acq-asl_run-02.json
|       |   |-- sub-219_ses-itbs_acq-asl_run-02.nii.gz
|       |   |-- sub-219_ses-itbs_task-rest_run-01_bold.json
|       |   |-- sub-219_ses-itbs_task-rest_run-01_bold.nii.gz
|       |   |-- sub-219_ses-itbs_task-rest_run-01_events.tsv
|       |   |-- sub-219_ses-itbs_task-rest_run-02_bold.json
|       |   |-- sub-219_ses-itbs_task-rest_run-02_bold.nii.gz
|       |   `-- sub-219_ses-itbs_task-rest_run-02_events.tsv
|       `-- sub-219_ses-itbs_scans.tsv
`-- task-rest_bold.json
```

Exploring Criteria

dicominfo.tsv contains a human readable version of seqinfo. Each column of data can be used as criteria for identifying the correct dicom image. We have already provided examples of using string types and numbers, but Booleans (True-False) and Tuples (immutable lists) are also available and examples of using these are provided below. To ensure that you are extracting the images you want, you need to be very careful about creating your initial convertall.py.

Before experimenting with criteria, I suggest you copy the Makefile example below into your project directory:

```
what:
      @echo "Make what? Try 'clean' or 'run'"

clean:
      rm -rf Nifti/sub-* Nifti/.heudiconv

run:
      docker run -it --rm -v ${PWD}:/base nipy/heudiconv:latest -d /base/Dicom/
↪{subject}/*/*.dcm -o /base/Nifti/ -f /base/Nifti/code/convertall.py -s 219 -ss itbs␣
↪-c dcm2niix -b --overwrite
```

Ensure that you used the naming strategy and modify path names for your environment if necessary. Now you can type
`make clean` to remove the .heudiconv and Nifti subject directory. Type `make run` to run the docker command.

Why Experiment?

- Criteria can be tricky. Ensure the NIFTI files you create are the correct ones (for example, not the derived
  or motion corrected if you didn't want that). In addition to looking at the images created (which tells you
  whether you have a fieldmap or T1 etc.), you should look at the dimensions of the image. For really tricky
  cases, download and install dcm2niix on your local machine and run it for a sequence of concern. Not only the
  dimensions, but the range of intensity values and the size of the image on disk should match for dcm2niix and
  heudiconv's convertall.py.

  - In the tutorial dataset, the *ASL_3D_tra_iso_22* and *Perfusion_Weighted_23* series are very similar, except
    the Perfusion weighted dataset is derived from the original ASL dataset, and the Perfusion dataset is
    smaller on disk (ASL image=~346 KB; Perfusion image=~128 KB). If you experiment with these two
    images, you can ensure that each was named as you expected.

  - Although Python does not require you to use parentheses while defining criteria, parentheses are a good
    idea. Parentheses will help ensure that complex criteria involving multiple logical operators `and, or,
    not` make sense and behave as expected.

Booleans

There are two columns containing Boolean True/False information in dicominfo.tsv, `is_derived` and
`is_motion_corrected`. To help you work with Booleans, you can experiment with the ASL and Perfusion
images mentioned above. They can be distinguished by the value in `is_derived`:

```python
if ('ASL_3D_tra_iso' == s.protocol_name):
            if (s.is_derived):
      info[asl_der].append(s.series_id)
  else:
      info[asl].append(s.series_id)
```

This conditional writes both images as different sequences. `s.is_derived` is `True` if the value in the
`is_derived` column is true, and the Perfusion image is written to NIFTI. Otherwise (else) `s.is_derived` is
not true and so the ASL image is written.

Of course, these particular examples depend on the creating of a key for each image:

```python
asl = create_key('sub-{subject}/{session}/func/sub-{subject}_{session}_acq-asl')
asl_der = create_key('sub-{subject}/{session}/func/sub-{subject}_{session}_acq-asl_der
↪')
```

To write only the original image and not the derived image, the following criteria would work:

```
if ('ASL_3D_tra_iso' == s.protocol_name):
            if (not s.is_derived):
      info[asl].append(s.series_id)
```

Tuples

Suppose you want to use the values in the field `image_type`? It is not a number or string or Boolean. How do we discover the data type of a column? You can add a statement like this `print(type(s.image_type))` to the for loop in section 2 of convertall.py. Then run convertall.py (preferably without any actual conversions) and you should see an output like this `<class 'tuple'>`. Here is an example of using a value from image_type as a criterion:

```
if ('ASL_3D_tra_iso' == s.protocol_name) and ('TTEST' in s.image_type):
   info[asl_der].append(s.series_id)
```

Note that this differs from testing for a string because we cannot test for any substring (e.g., 'TEST' would not work). String tests will not work on a tuple datatype.

Note: *image_type* is described in the DICOM specification

---

reproin.py

If you don't want to modify a Python file as we do for *convertall.py*, an alternative is to name your image sequences at the scanner using the *reproin* naming scheme. Take some time getting the scanner protocol right, because it is the critical job for running *reproin*. Then a single Docker command converts your DICOMS to the BIDS data structure. There are more details about *reproin* in the *Links* section above.

Walkthrough of Reproin Run

Download this phantom dataset: dicoms_reproin_UA.zip generated here at the University of Arizona on our Siemens Skyra 3T with Syngo MR VE11c software on 2018_02_08. This is a simple reproin-compliant dataset without sessions. Derived dwi images (ADC, FA etc.) that the scanner produced were removed.

We'll assume you already have Docker installed and have downloaded HeuDiConv as described above. Create a new directory, lets call it *ReproUA*. Change directory to *ReproUA* and create a subdirectory called *Dicom*. Unzip dicoms_reproin_UA.zip in the *Dicom* directory. You should have a directory structure like this:

```
ReproUA
`-- Dicom
    |-- 001
```

From the *ReproUA* directory, run this Docker command:

```
docker run --rm -it -v ${PWD}:/base nipy/heudiconv:latest -f reproin --bids  -o /base/
↪Nifti --files /base/Dicom/001
```

That's it. Below we'll unpack what happened.

At the Scanner

Here is our phantom dataset displayed in the scanner dot cockpit. The directory structure is defined at the top: *Patterson >> Coben >> Patient*

- *Region = Patterson*
- *Exam = Coben*
- *Program = Patient*
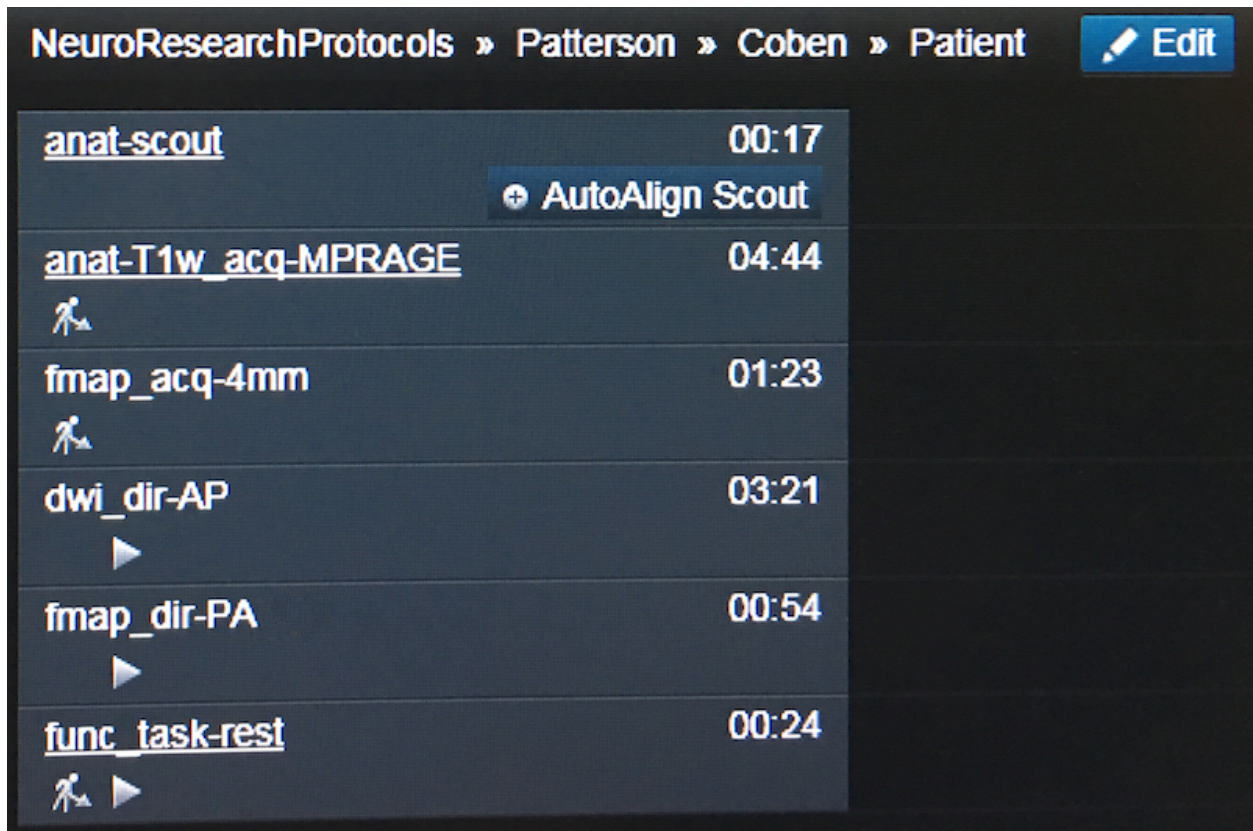
Output Directory Structure

Reproin produces a hierarchy of BIDS directories like this:

```
.
`-- Patterson
    `-- Coben
        |-- sourcedata
        |   `-- sub-001
        |       |-- anat
        |       |-- dwi
        |       |-- fmap
        |       `-- func
        `-- sub-001
            |-- anat
            |-- dwi
```

```
        |-- fmap
        `-- func
```

- The dataset is nested under two levels in the output directory: *Region* (Patterson) and *Exam* (Coben). *Tree* is reserved for other purposes at the UA research scanner.

- Although the Program *Patient* is not visible in the output hierarchy, it is important. If you have separate sessions, then each session should have its own Program name.

- **sourcedata** contains tarred gzipped (tgz) sets of DICOM images corresponding to each NIFTI image.

- **sub-001** contains NIFTI and JSON pairs in addition to the metadata files BIDS wants.

File Names

For both BIDS and *reproin*, names are composed of an ordered series of key-value pairs. Each key and its value is joined with a dash - (e.g., `acq-MPRAGE`, `dir-AP`). These key-value pairs are joined to other key-value pairs with underscores _. The exception is the modality label, which is discussed more below.

Reproin scanner sequence names are simplified relative to the final BIDS output and generally conform to this scheme (but consult the reference for additional options): `sequence type-modality label _ session-session name _ task-task name _ acquisition-acquisition detail _ run-run number _ direction-direction label`.

e.g., *func-bold_ses-pre_task-faces_acq-1mm_run-01_dir-AP*.

- Each sequence name begins with the seqtype key. The seqtype key is the name of the BIDS directory where the sequence belongs, e.g., `anat`, `dwi`, `fmap` or `func`.

- The seqtype key is optionally followed by a dash - and a modality label value (e.g., `anat-scout` or `anat-T2W`). Often, the modality label is not needed because there is a predictable default for most seqtypes:

    - For **anat** the default modality is `T1W`. Thus a sequence named `anat` will have the same output BIDS files as a sequence named `anat-T1w`, e.g., `sub-001_T1w.nii.gz` in the **anat** directory.

    - For **fmap** the default modality is `epi`. Thus `fmap_dir-PA` will have the same output as `fmap-epi_dir-PA`, e.g., `sub-001_dir-PA_epi.nii.gz` in the **fmap** directory.

    - For **func** the default modality is `bold`. Thus, `func-bold_task-rest` will have the same output as `func_task-rest`, e.g., `sub-001_task-rest_bold.nii.gz` in the **func** directory.

    - *Reproin* gets the subject number from the DICOM metadata.

- If you have multiple sessions, the session name does not need to be included in every sequence name in the program (i.e., Program= *Patient* level mentioned above). Instead, the session can be added to a single sequence name, usually the scout (localizer) sequence e.g. `anat-scout_ses-pre`, and *reproin* will propagate the session information to the other sequence names in the *Program*. Interestingly, *reproin* does not add the localizer to your Nifti output.

- When our scanner exports the DICOM sequences, all dashes are removed. But don't worry, *reproin* handles this just fine.

- In the UA phantom reproin data, the subject was named `01`. Horos reports the subject number as `01` but exports the DICOMS into a directory `001`. If the data are copied to an external drive at the scanner, then the subject number is reported as `001_001` and the images are `*.IMA` instead of `*.dcm`. *Reproin* does not care, it handles all of this gracefully. Your output tree (excluding sourcedata) should look like this:

```
.
|-- CHANGES
|-- README
|-- dataset_description.json
|-- participants.tsv
|-- sub-001
|   |-- anat
|   |   |-- sub-001_acq-MPRAGE_T1w.json
|   |   `-- sub-001_acq-MPRAGE_T1w.nii.gz
|   |-- dwi
|   |   |-- sub-001_dir-AP_dwi.bval
|   |   |-- sub-001_dir-AP_dwi.bvec
|   |   |-- sub-001_dir-AP_dwi.json
|   |   `-- sub-001_dir-AP_dwi.nii.gz
|   |-- fmap
|   |   |-- sub-001_acq-4mm_magnitude1.json
|   |   |-- sub-001_acq-4mm_magnitude1.nii.gz
|   |   |-- sub-001_acq-4mm_magnitude2.json
|   |   |-- sub-001_acq-4mm_magnitude2.nii.gz
|   |   |-- sub-001_acq-4mm_phasediff.json
|   |   |-- sub-001_acq-4mm_phasediff.nii.gz
|   |   |-- sub-001_dir-PA_epi.json
|   |   `-- sub-001_dir-PA_epi.nii.gz
|   |-- func
|   |   |-- sub-001_task-rest_bold.json
|   |   |-- sub-001_task-rest_bold.nii.gz
|   |   `-- sub-001_task-rest_events.tsv
|   `-- sub-001_scans.tsv
`-- task-rest_bold.json
```

- Note that despite all the the different subject names (e.g., `01`, `001` and `001_001`), the subject is labeled `sub-001`.

2.1.2 Introduction

This page provides detailed instructions, examples and scripts for using Singularity containers that I provide on the HPC for the neuroimaging community. Some discussion of Docker containers is also provided. As always, check **Date Updated** above to ensure you are not getting stale information. My preference is to point to official documentation when I can find it and think it is useful. If you have found documentation you believe I should consider linking to, please let me know (dkp @ email.arizona.edu).

BIDS Containers are BIG and Resource Intensive

- The containers we use for neuroimage processing tend to be large and resource intensive. That brings its own set of problems.

- For **Docker** on the mac, you need to check **Preferences Disk** to allow Docker plenty of room to store the containers it builds (several containers can easily require 50-100 GB of space). In addition, check **Preferences Advanced** to make sure you give Docker plenty of resources to run tools like Freesurfer. For example, I allocate 8 CPUs and 32 GB of memory to Docker).

- In theory, it is possible to move your Docker storage to an external disk. In my experience this is flaky and results in slowness and lots of crashing (early 2019). Of course, it could be better in the future.

- The implications of container size for **Singularity** containers are explored *here*.

- Below I provide information about neuroimaging Singularity containers I maintain on the U of A HPC. You are free to use these if you have an HPC account. In addition to providing containers, I also provide example scripts for running those containers. You'll have to copy and modify those scripts for your own directories and group name. A detailed walk-through is provided here *BET*.

Create Subdirectories under Derivatives for your Outputs

In general, you want to put results of running these containers into a derivatives directory. fMRIPrep creates subdirectories under derivatives, which seems like a good idea as it keeps the outputs of different containers separated and it does not annoy the bids validator. At the time of this writing, 02/27/2020, the standards for naming in the derivatives directory have not been finalized.

Note: It is a good idea to create the derivatives directory before you run your Docker or Singularity container. Sometimes the containerized app looks for a directory and fails if it does not find it.

2.1.3 BET

This is a small neuroimaging container (under 500 MB), which runs quickly. This walk-through will provide you with experience working at the unix commandline, transferring data, running interactive and batch mode processes on the HPC, building a Singularity container from a Docker container, and running a bids-compliant workflow with Singularity on the HPC. This assumes you have an HPC account and are familiar with the material on the *HPC page* page. You will also need some experience with the *Unix comand line*.

Login to OOD and Get Ready

- Open a File Explorer window in your home directory (Files Home Directory).
- From the File Explorer window, select `Open in Terminal` (at the top, 2nd from the left) and choose Ocelote.

Use the Dataset on the HPC

Copy the tutorial data to your home directory: The first command below, `cd`, ensures you are actually in your home directory. The second command copies the dataset to your home directory:

```
cd
cp /extra/dkp/tutorial_dataset/bids_data.zip .
```

Optional: Try Data Transfer

Download bids_data.zip. The dataset is too big to upload with the OOD `upload` button. Use `scp` instead, e.g. (note you will need to use your username instead of `dkp`):

```
scp -v bids_data.zip dkp@filexfer.hpc.arizona.edu:~
```

Prepare the Dataset

The first command below unzips the dataset. The second command changes directories so you are in the unzipped dataset folder. The third command creates a derivatives subdirectory for your output. The last command changes your directory again, so you are up one level from `bids_data`:

```
unzip bids_data.zip
cd bids_data
mkdir derivatives
cd ..
```

Build the BET Singularity Container

Copy interact_small.sh from /extra/dkp/Scripts:

```
cp /extra/dkp/Scripts/interact_small.sh .
```

Start an interactive session:

```
./interact_small.sh
```

interact_small.sh tells you what groups you are in, so you can start it correctly. I am in group `dkp`, so the following works for me. It may take several minutes to start your interactive job depending on how busy the HPC is:

```
./interact_small.sh dkp
```

Once you have an interactive prompt, you can build the Singularity container. First, you have to load the Singularity module so the HPC will understand Singularity commands you issue. Second, you can build the container by pulling it from dockerhub:

```
module load singularity
singularity build bet.sif docker://bids/example
```

Note: If you have any trouble with this step, you can use /extra/dkp/singularity-images/bet.sif instead. You are welcome to copy it, but you can also use it without copying it.

Run BET with Singularity

You should be in interactive mode and have the Singularity module loaded. Determine whether the Singularity module is loaded:

```
module list
```

You should see something like this (we are interested in #3):

```
Currently Loaded Modulefiles:
1) pbspro/19.2.4   2) gcc/6.1.0(default)   3) singularity/3/3.5.3
```

Run Singularity on the dataset:

```
singularity run --cleanenv --bind bids_data:/data ./bet.sif  /data /data/derivatives
→participant --participant_label 1012
```

If you do not have the bet.sif container in your home directory, you can use the one in /extra/dkp/singularity-images:

```
singularity run --cleanenv --bind bids_data:/data /extra/dkp/singularity-images/bet.
↪sif  /data /data/derivatives participant --participant_label 1012
```

If Singularity runs properly it creates `sub-1012_brain.nii.gz` in the bids_data/derivatives directory. Confirm that it worked:

```
cd bids_data
ls derivatives
```

Provided that worked, we can run the group-level BIDS command. Let's try it from the bids_data directory this time:

```
singularity run --cleanenv --bind ${PWD}:/data ../bet.sif /data /data/derivatives␣
↪group
```

That should create `avg_brain_size.txt` in the derivatives directory.

Understanding the Singularity Command

Singularity takes a number of options. So far you've seen `build` and `run`. `build` creates the sif file. `run` uses that file to perform some processing.

- `--cleanenv` prevents conflicts between libraries outside the container and libraries inside the container; although sometimes the container runs fine without `--cleanenv`, it is generally a good idea to include it.

- `--bind` Singularity (like Docker) has a concept of what is inside the container and what is outside the container. The BIDS standard requires that certain directories exist on the inside of every BIDS container, e.g., /data (or sometimes /bids_dataset) and /outputs. You must bind your preferred directory outside the container to these internal directories. Order is important (outside:inside). Here are our two examples: `--bind bids_data:/data` or `--bind ${PWD}:/data`

- What container are we running? You must provide the unix path to the container. There are three examples here:

    - `./bet.sif` assumes that bet.sif is in the same directory where you are running the singularity command.

    - `/extra/dkp/singularity-images/bet.sif` provides the path to bet.sif in /extra/dkp/singularity-images.

    - `../bet.sif` says the container is up one directory level from where you are running the singularity command.

- BIDS requires that we list input and output directories. This is relative to the bind statement that defines the directory on the outside corresponding to `/data` on the inside. Thus `/data/derivatives` will correctly find our derivatives directory outside the container. This is the same for Docker containers.

- Finally, further BIDS options are specified just like they would be for the corresponding Docker runs.

Running a Batch Job

Batch jobs, like interactive mode, use your allocated time. Copy runbet.sh from `/extra/dkp/Scripts`:

```
cp /extra/dkp/Scripts/runbet.sh .
```

The script consists of two sections. The first section specifies the resources you need to run the script. All the scripts I make available to you have pretty good estimates of the time and resources required.

The second part of the script is a standard bash script. It defines a variable `Subject` and calls Singularity.

Open the script with the *editor*, because you will need to modify it to specify your group name on line 13. Change the group name from `group_list=dkp` to your own group name, e.g.,:

```
#PBS -W group_list=akielar
```

In addition, on line 40 put in your email address instead of mine:

```
#PBS -M joe@.arizona.edu
```

Save the script. We will pass the subject variable to qsub using `-v sub=1012`:

```
qsub -v sub=1012 runbet.sh
```

Look in the active jobs window to see if your job is queued. It runs very quickly so it may complete before you have a chance to see it. When it finishes, it creates a text log (e.g., `bet.o3117511`) using the name you specified on line 10 of the PBS script. The job submission system should also send you an email from **root** telling you the job is complete. See below. `Exit status=0` means the job completed correctly. The job used 11 seconds of walltime and 5 seconds of cpu time:

```
PBS Job Id: 3117556.head1.cm.cluster
Job Name:   bet
Execution terminated
Exit_status=0
resources_used.cpupercent=30
resources_used.cput=00:00:05
resources_used.mem=77428kb
resources_used.ncpus=1
resources_used.vmem=746916kb
resources_used.walltime=00:00:11
```

Other Batch Scripts

Other scripts are available on bitbucket and in /extra/dkp/Scripts. Some of them create additional directory structure or run as arrays. Read more about *qsub*.

In addition to altering the group and the email in the PBS portion of the scripts, many of the scripts export a top level directory `STUFF`. This can be very useful because you'll want to place some files and directories outside of your BIDS data directory. Change `STUFF` to point to your directory, instead of `/extra/dkp`, e.g.,:

```
export STUFF=/extra/akielar
```

- Change the `MRIS` variable if appropriate as well. This is the directory containing your BIDS compliant subject directories.

2.1.4 BIP

BIP (Bidirectional Iterative Parcellation), runs FSL dwi processing with BedpostX and Probtrackx2. The twist is that BIP runs Probtrackx2 iteratively until the size of the connected grey matter regions stabilizes. This provides a unique and useful characterization of connectivity (the locations and volumes of the connected grey matter regions) not available with other solutions.

Patterson, D. K., Van Petten, C., Beeson, P., Rapcsak, S. Z., & Plante, E. (2014). Bidirectional iterative parcellation of diffusion weighted imaging data: separating cortical regions connected by the arcuate fasciculus and extreme capsule. NeuroImage, 102 Pt 2, 704–716. http://doi.org/10.1016/j.neuroimage.2014.08.032

A detailed description of how to run BIP is availabe as a Readme on the bipbids Bitbucket site. BIP runs in three stages: setup, prep and bip. **setup** prepares the T1w and dwi images. **prep** runs eddy, dtifit and bedpostX. **bip** does the iterating for the selected tracts.

Docker

Any of these steps can be run with a local Docker container: diannepat/bip2. Run `docker pull diannepat/ bip2` to get the container and download the helpful bash script bip_wrap.sh.

Singularity

To take advantage of GPU processing for the prep and bip steps, you should run the Singularity container.

You can build the Singularity container from the Singularity_bip recipe . See *Build Singularity Containers from Recipes*.

For more about the GPU code, see Bedpostx_GPU (BedpostX_gpu runs in 5 minutes instead of 12-24 hours for Bed-postX) and Probtrackx_GPU (200x faster). The result of running Probtrackx_GPU is slightly different than running probtrackx so don't mix results from the regular and GPU versions.

A Singularity container is available on the HPC: `/extra/dkp/singularity-images/bip2.sif`. Several example scripts that facilitate running the different stages of BIP can be found in `/extra/dkp/Scripts` and on Bitbucket). Here are examples:

```
# This script calls prep on a single subject that you pass to qsub
qsub -v sub=1012 runbip2prep.sh

# This script calls prep on an array of subjects
qsubr Scripts/arraybip2prep.sh SubjectLists/plante181-252.subjects

# This script runs a single subject and tract that you pass to qsub
qsub -v "sub=1012, tract=arc_l" runbip2bip_1.sh

# This script also run an array job.
# It includes several calls to singularity, each for a different tract:
qsubr Scripts/arraybip2bip_2.sh SubjectLists/plante253-294.subjects
```

- As with all of the scripts I've made available, you'll want to copy them into your own directory, modify the STUFF and MRIS directory, the associated group name and the email address.

- In these GPU scripts, you'll note that 28 CPUs are requested: `#PBS -l select=1:ncpus=28:mem=56gb:ngpus=1`. This is a whole node on Ocelote and is necessary in order to use the GPU processing capabilities. In addition, you'll notice that the cuda 10 module is loaded: `module load cuda10.0`. Cuda is required for GPU processing. Any Cuda version later than cuda80 is backward compatible, so feel free to use the latest and greatest Cuda version available when you run.

- Example code to run on el gato's GPUs is commented out but available in the scripts. There are two differences between El Gato and Ocelote with regard to the GPUs: (1) el gato has 16 nodes per GPU instead of 28. (2) el gato uses `module load cuda10` instead of `module load cuda10.0`.

- Just like bip_wrap.sh, these scripts assume derivative data will go into subdirectories of derivatives: `fsl_anat_proc`, `fsl_dwi_proc` and `bip`. These subdirectories are the same ones used by dwiqc, de-scribed below, so if you've run one, you will not have to worry about creating redundant output or wasting time repeating the same steps.

> **Warning:** BedpostX_gpu will create all the appropriate output files even when it fails to run correctly! Clues are: The files (dyads, mean, merged etc) are too small. The logs in the dwi.bedpostX will report the failure. ProbtrackX fails to run correctly when BedpostX has failed.

Job Times

prep takes roughly 3 hours of walltime. bip takes 5-10 minutes for a single tract and single subject. This varies by tract.

2.1.5 DWIQC

DWIQC uses FSL 6.X to run topup, eddy and the eddy quality control metrics. It relies on Siemens fieldmaps and several text files to define the parameters for running.

Docker

DWIQC can be run with a local Docker container: diannepat/dwiqc. Run `docker pull diannepat/dwiqc` to get the container and download the helpful bash script dwiqc_wrap.sh.

Singularity

A singularity image is available on the HPC: `/extra/dkp/singularity-images/dwiqc.sif`, along with scripts that facilitate running it (/extra/dkp/Scripts) at the participant: rundwiqc.sh and group rundwiqc_group.sh levels, and as an array job at the participant level arraydwiqc.sh. Each script has appropriate parameters for cpu, memory and walltime. The dwiqc Readme provides more information.

Here are examples of using the script on the HPC:

```
# Here we run a single subject
qsub -v sub=1012 Scripts/rundwiqc.sh

# Here we run an array job
qsubr Scripts/arraydwiqcPB.sh SubjectLists/plante078-084.subjects
```

Job Times

DWIqc on the HPC takes about 2 hours to run. There is probably some variation based on head size and number of directions.

2.1.6 fMRIPrep

There is lots of online documentation for fMRIPrep.

Note: You need to download a Freesurfer license and make sure your container knows where it is. See The Freesurfer License

---

Note: Be careful to name sessions with dashes and not underscores. That is `itbs-pre` will work fine, but `itbs_pre` will cause you pain in later processing.

You can rerun fMRIPrep and it'll find its old output and avoid repeating steps, especially if you have created the -w work directory (the runfmriprep.sh script does this). So, it is not the end of the world if you underestimate the time needed to run.

Determine which **version** of fMRIPrep is in the Singularity container (This is especially useful for fmriprep as it changes frequently and several versions are available in `/extra/dkp/singularity-images`):

```
singularity run --cleanenv /extra/dkp/singularity-images/fmriprep.sif --version
```

A usable Singularity job script is available here: runfmriprep.sh. This should be easy to modify for your own purposes.

- fMRIPrep needs to know where your freesurfer license is. Ensure the singularity command points to your license and that it is outside of your BIDS directory:

```
--fs-license-file ${STUFF}/license.txt
```

- Create a work directory for fMRIPrep. This is where it will store intermediate steps it can use if it has to start over. Like the freesurfer license, this directory should not be in your BIDS directory:

```
-w ${STUFF}/fmriprep_work
```

- BIDS singularity images are big and take some effort to build (Gory details are in *Building Large Containers* if you want to do it yourself).

- Currently, the singularity command in this script points to the fMRIPrep singularity container in `/extra/dkp/singularity-images`:

```
/extra/dkp/singularity-images/fmriprep.sif
```

- Permissions should allow you to use containers in this directory freely. If my version of the containers is satisfactory for you, then you do not need to replicate them in your own directory. I am hoping we'll have a shared community place for these images at some point (other than my directory).

- You do not NEED to change any other arguments. `--stop-on-first-crash` is a good idea. You may wish to test with the reduced version that does not run reconall. It is currently commented out, but it'll succeed or fail faster than the default call.

- Once you are satified with the changes you've made to your script, run your copy of the script like this:

```
qsub -v sub=1012 Scripts/runfmriprep.sh
```

- qsub needs the path to your script (in my case, it is in the `Scripts` directory).

- When the job finishes (or crashes), it'll leave behind a text log, e.g., `fmriprep_extra.o2252098`. You can view a log of the job in the **Jobs** dropdown on the OOD dashboard.

- Read this log with `Edit` in OOD or `cat` at the command line. It may suggest how many resources you should have allocated to the job (scroll to the bottom of the job file). This can tell you whether you have vastly over or under-estimated. In addition, it'll provide a log of what the container did, which may help you debug.

- See *BIDS containers* for more detail about individual containers.

Job Times

Official job times

2.1.7 MRIQC

MRIQC is from the Poldracklab, just like fmriprep. MRIQC runs quickly and produces nice reports that can alert you to data quality problems (mostly movement, but a few other issues).

Docker

The mriqc Docker container is available on dockerhub:

```
docker pull poldracklab/mriqc
```

A wrapper script for the Docker container is also available, mriqc_wrap.sh.

Singularity

MRIQC is available on the HPC: `/extra/dkp/singularity-images/mriqc.sif`, along with two scripts that facilitate running it at the participant: `/extra/dkp/Scripts/runmriqc.sh` and group `/extra/dkp/Scripts/runmriqc_group.sh` levels. Here is an example of using the script on the HPC. We pass in the subject number:

```
qsub -v sub=1012 Scripts/runmriqc.sh
```

An participant level mriqc run with one T1w anatomical image and one fMRI file took about 25 minutes of walltime with 2 cpus. A group run with a single subject took just under 4 minutes with 2 cpus.

The HTML reports output by MRIQC can be viewed on OOD by selecting `View`.

Job Times

MRIQC job times vary by the number of files to be processed. Examples on the HPC are 22 minutes for a T1w image only; 1+ hours for a T1w image and 4 fMRI images.

2.1.8 MRtrix3_connectome

MRtrix3_connectome facilitates running the MRtrix software, which processes DWI images to create a connectome. To determine which version of MRtrix3_connectome you have, you can run the following command on Docker:

```
docker run --rm bids/mrtrix3_connectome -v
```

Or, on the HPC, you can run the equivalent Singularity command (this is quick, you do not need to submit a job):

```
module load singularity
singularity run mrtrix3_connectome.sif -v
```

Singularity

- Here's a script for running MRtrix3_connectome on the HPC using the HCP (Human Connectome) atlas runmrtrix3_hcp.sh

- And, here's a script for running MRtrix3_connectome on the HPC using the Desikan atlas runmrtrix3_desikan.sh

- Finally, below is a call to `qsub` to run the desikan script on a particular subject.

- Note that you should copy the script to your own area on the HPC and change the group_list in the script from dkp to your group. The time and CPU requirements should be roughly correct:

```
qsub -v sub=1012 runmrtrix3_desikan.sh
```

Note: As of 04/06/2020: Read *how operating systems interact with singularity* before attempting to run the MRtrix3_connectome container. You have to run mrtrix3_connectome on ElGato or using a special argument in your qsub statement on Ocelote to set CentOS to version 7.

Job Times

Jobs were run on ElGato, because Ocelote's operating system was too old to run this MRtrix3_connectome container at the time of this writing. See the section on *Host OS and Singularity Interactions*

Test Data DWI image with 64 directions, 2 mm isotropic voxels and 10 B0 images. T1w image with 1 mm isotropic voxels.

- **desikan Parcellation** 16 cores, 8 GB of RAM and 13 hours walltime. Working sample script: runmrtrix3_desikan.sh

- **hcpmmp1 Parcellation** 16 cores 13 GB of RAM, and 22 hours walltime. Working sample script: runmrtrix3_hcp.sh

Learn about this two-pronged project which describes (1) a standard file and directory naming structure for neuroimaging datasets, and (2) containerized apps that take assume this naming structure to great advantage. Below I provide useful links, including descriptions of software we've actually tried here at the U of A and related gotchas and tips.

2.2 Main Links

- The neuroimaging data structure
- The neuroimaging docker containers
- BIDS tutorials
- BIDS Read-the-Docs

2.2.1 BIDS compatible datasets

- Available dataset descriptions
- Downloadable examples

2.3 Relevant Papers

- Gorgolewski, K. J., Auer, T., Calhoun, V. D., Craddock, R. C., Das, S., Duff, E. P., et al. (2016). The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. Scientific Data, 3, 160044–9. http://doi.org/10.1038/sdata.2016.44 https://www.nature.com/articles/sdata201644

- Gorgolewski, K. J., Alfaro-Almagro, F., Auer, T., Bellec, P., Capotă, M., Chakravarty, M. M., et al. (2017). BIDS apps: Improving ease of use, accessibility, and reproducibility of neuroimaging data analysis methods. PLoS Computational Biology, 13(3), e1005209–16. http://doi.org/10.1371/journal.pcbi.1005209 http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1005209

- van Mourik, T., Snoek, L., Knapen, T., & Norris, D. (2017). Porcupine: a visual pipeline tool for neuroimaging analysis, 1–10. http://doi.org/10.1101/187344 https://www.biorxiv.org/content/early/2017/10/11/187344

2.4 Educational and Related Materials

- The BIDS folder hierarchy

- Metadata file formats

- The official tutorial

- Data for the official tutorial: Download ds005.tar here

- Join BIDS google groups

- An introduction to Docker for BIDS users

- Video of docker-bids workshop by Chris Gorgolewski (links to his other videos on right)

- Dataflows for BIDS

- BIDS templates

- A presentation on the BIDS data format and HeuDiConv for DICOM conversion

- fmriprep documentation

    - fmriprep and containers

- Porcupine software for auto-generating your pipeline (see paper above)

2.5 Creating BIDS Datasets

Tools available for creating BIDS datasets focus on converting DICOMS to NIFTI and creating the accompanying json sidecar files for each NIFTI image. If you don't have the original images off the scanner, there are not currently any good solutions for converting your data to BIDS. Part of the reason you should start with the DICOM (i.e., dcm or IMA images here at the U of A), is that there is considerable information in the DICOM headers that does not get transferred to the NIFTI headers. If you convert the DICOMs to BIDS, then this extra (deidentified) information gets saved in the json sidecar. The defacto tool for conversion is dcm2niix by Chris Rorden: download dcm2niix and read more about dcm2niix. dcm2niix produces the BIDS format json sidecar files, handles dcm and IMA files, and correctly saves slice timing for multiband data (which is not preserved in the NIFTI header).

Note: Neither dcm2niix nor *HeuDiConv* will name your output files in a bids compliant way by default. You must look at the BIDS specification and figure out the naming. That said, I try to provide examples and guidance for UofA data on the *HeuDiConv* page.

Note: The BIDS specification is under active development. In general, if your data is named in a compliant way today, it'll still be okay tomorrow. However, tomorrow may bring additional options.

2.5.1 bids-validator

Author/Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu

Date Created: 2018_12_14

Date Updated: 2019_07_15

Tags: BIDS, standards

OS: Any

Introduction

BIDS datasets should conform to standards described in the evolving Brain Imaging Data Structure (BIDS) Specification. When you run a BIDS app, the data structure is checked to ensure it conforms (though BIDS apps normally allow you to turn the checking off).

bids-validator online

You can check your data structure using the online bids-validator (which works in Google Chrome and Mozilla Firefox). At the top of the interface where it says *Choose File*, select the parent directory containing subject subdirectories. There must be at least one subject directory for the validator to run correctly. **Don't worry!! You are not uploading your files**, the validator is looking at the directory and image names and the json file names and contents.

Here the error is that any image and json file in a session directory must include the session label in its name. Error messages from the validator can be very helpful, but are sometimes nonsensical (e.g., claiming that you can't use hyphens or underscores or that a field does not exist in a json file, even though you know it does exist). If nonsensical, you should should not take the reported infractions at face value, they likely reflect a different problem.

- Review the naming carefully and ensure that you don't have any directories containing non-bids data in the path.

- If you use sessions, then every image under a session directory must be labelled with the session label.

- Also click *Download error log for Nifti* (see bottom of bids-validator page) so that you can submit the log to the BIDS group for help.

Phasediff images

Some BIDS apps (e.g., MRtrix3_connectome) expect the phasediff .json sidecar to include *EchoTime1* and *EchoTime2* fields. dcm2niix does not currently provide those two fields because of the way Siemens has stored the dicom information. The phasediff image gets exported with a single *EchoTime* field (which actually corresponds to *EchoTime2*). You can find *EchoTime1* in the Magnitude1 image (again, just labelled *EchoTime*). However, at this point you have to manually alter the phasediff json file to include the correct fields and values for apps that require it (e.g., MRtrix3_connectome). A single .json file that contains the correct information can also be added at the level of the parent directory.

BIDS apps with narrow assumptions

Sometimes BIDS apps assume naming conventions which are not required for bids validation. For example, the app may expect particular fields in the json file or a particular naming convention for the files rather than handling the variety of possibilities acceptable to the validator.

In addition, certain kinds of processing (like running FSL topup) rely pretty exclusively on MRI data from a particular manufacturer (Siemens). So, despite the goal of using any BIDS app on any correctly organized dataset, there may still be problems for a particular app with special assumptions.

Creating a .bidsignore

If you are including files or directories that are not handled by the official bids specification, then the bids-validator will produce errors. To address this issue, you can create a small hidden file at the same level as your subject directories. .bidsignore uses the syntax of .gitignore

For example, this tells the bids-validator to ignore any files in any directories that contain *asl* or *tse* in the names:

```
**/*asl*
**/*tse*
```

This is useful since (as of this writing) the *asl* and *tse* filetypes are not yet handled by BIDS. The .bidsignore file is directly under the directory *Nifti* in the example below.

Example

The directory structure below is fully bids compliant. It does not have to be called Nifti, but that is the default used by *HeuDiConv*:

```
Nifti
|-- .bidsignore
|-- .heudiconv
|   `-- 219
|       |-- info
|       |   |-- 219.auto.txt
|       |   |-- 219.edit.txt
|       |   |-- convertall.py
|       |   |-- convertall_old.py
|       |   |-- dicominfo_wrong_name.tsv
|       |   `-- filegroup.json
|       `-- ses-itbs
|           `-- info
|               |-- 219_ses-itbs.auto.txt
|               |-- 219_ses-itbs.edit.txt
|               |-- convertall.py
|               |-- dicominfo_ses-itbs.tsv
|               `-- filegroup_ses-itbs.json
|-- CHANGES
|-- README
|-- code
|   |-- __pycache__
|   |   `-- convertall.cpython-36.pyc
|   `-- convertall.py
|-- dataset_description.json
|-- participants.tsv
```

(continues on next page)

```
|-- sub-219
|   `-- ses-itbs
|       |-- anat
|       |   |-- sub-219_ses-itbs_T1w.json
|       |   |-- sub-219_ses-itbs_acq-tse_T2w1.json
|       |   |-- sub-219_ses-itbs_acq-tse_T2w1.nii.gz
|       |   |-- sub-219_ses-itbs_acq-tse_T2w2.json
|       |   |-- sub-219_ses-itbs_acq-tse_T2w2.nii.gz
|       |   `-- sub-219_ses-itbs_T1w.nii.gz
|       |-- dwi
|       |   |-- sub-219_ses-itbs_acq-AP_dwi.bval
|       |   |-- sub-219_ses-itbs_acq-AP_dwi.bvec
|       |   |-- sub-219_ses-itbs_acq-AP_dwi.json
|       |   `-- sub-219_ses-itbs_acq-AP_dwi.nii.gz
|       |-- fmap
|       |   |-- sub-219_ses-itbs_dir-PA_epi.json
|       |   |-- sub-219_ses-itbs_dir-PA_epi.nii.gz
|       |   |-- sub-219_ses-itbs_magnitude1.json
|       |   |-- sub-219_ses-itbs_magnitude1.nii.gz
|       |   |-- sub-219_ses-itbs_magnitude2.json
|       |   |-- sub-219_ses-itbs_magnitude2.nii.gz
|       |   |-- sub-219_ses-itbs_phasediff.json
|       |   `-- sub-219_ses-itbs_phasediff.nii.gz
|       |-- func
|       |   |-- sub-219_ses-itbs_acq-asl_run-01.json
|       |   |-- sub-219_ses-itbs_acq-asl_run-01.nii.gz
|       |   |-- sub-219_ses-itbs_acq-asl_run-02.json
|       |   |-- sub-219_ses-itbs_acq-asl_run-02.nii.gz
|       |   |-- sub-219_ses-itbs_task-rest_run-01_bold.json
|       |   |-- sub-219_ses-itbs_task-rest_run-01_bold.nii.gz
|       |   |-- sub-219_ses-itbs_task-rest_run-01_events.tsv
|       |   |-- sub-219_ses-itbs_task-rest_run-02_bold.json
|       |   |-- sub-219_ses-itbs_task-rest_run-02_bold.nii.gz
|       |   `-- sub-219_ses-itbs_task-rest_run-02_events.tsv
|       `-- sub-219_ses-itbs_scans.tsv
`-- task-rest_bold.json
```

Bookmarks

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2018_09_09
Date Updated: 2019_10_31
Tags:

Also see the *Glossary* page.

3.1 Brain Atlases

- Brain Info is designed to help you identify structures in the brain.

- CORTICAL ATLAS PARCELLATIONS IN MNI SPACE

- NIF, the Neuroscience Information framework, is a dynamic inventory of web-based neuroscience resources.

- OBART The Online Brain Atlas Reconciliation Tool (OBART) aims to provide a quantitative solution to the so-called neuroanatomical nomenclature problem by comparing overlap relations between regions defined as spatial entities in different digital human brain atlases.

3.2 Computational Resources

- BIDS Brain Imaging Data Structure links

- Cyverse Cyberinfrastructure supporting online customizable virtual machines (Atmosphere), containerized apps (Discovery Environment) and data storage (Data Commons). You must request access to each service separately.

- Docker Containers

- GIT Repositories An excellent set of tutorials introducing the concept of code repositories and how to use them. Examples use Gitkraken which is a graphical git client available for free to academics.

- HPC Documentation University of Arizona High Performance Computing Documentation
    - HPC Account Creation
    - OOD Open On Demand Dashboard to interact with HPC resources.
    - HPC Introduction
- Jupyter Org A free online unix terminal and Jupyter Notebook server.
- OpenNeuro A site that implements BIDS apps in an easily accessible gui interface. The catch is that you have to make your data public after a couple of years.
- REDCap A fully IRB-approved HIPAA-compliant web database application freely available to U of A researchers for capturing and querying data (e.g., we could have potential participants fill out their MRI screening here).
- Research Bazaar Unix Introduction
- SCIF The Scientific Filesystem
- Software Carpentry
- Software and Data Carpentry: University of Arizona
- Singularity Containers

3.3 fMRI Databases

- The Brede Database allows you to look up a brain region and discover what tasks activate it.
- Neurosynth is a platform that automatically synthesizes fMRI results across many studies.
- OpenNeuro See *Computational Resources* above. This is also a site to get publicly available BIDS compliant datasets.

3.4 Instructional: Neuroimaging Software and Concepts

- Andy's Brain Book Andrew Jahn's Read The Docs pages
- Andy's Brain Blog Online tutorials and videos on a variety of topics: Afni, Eprime, Freesurfer, FSL, FSLeyes, Unix
- FMRIF Summer 2017 Course Materials fMRI powerpoints and video presentations
- GIFT Resources related to the GIFT ICA Matlab processing toolbox: presentations, a manual, sample data, a short video etc.
- Standard fMRI Analysis Tutorials Rajeed Raizada provides some excellent interactive Matlab tutorials for MVPA (multi-voxel pattern analysis) and SPM fMRI analysis.
- Resting State Analyis: How To Online resource from Jeanette Mumford et al.

3.5 MRI Concepts

- MRI Questions provides articles explaining how MRI works.
- MR-Tip is a glossary of MRI-related terms.

3.6 Neuroimaging Tools

- NITRC, the Neuroimaging Tools and Resource Collaboratory, is a huge database of neuroimaging tools and datasets.

3.7 Other

- The Massive Memory Database is a huge database of images useful for building visual stimulus sets. Images are organized into useful categories, like *state pairs*, *exemplar pairs* etc.
- Open Brain Consent A discussion of how to word your consent form to allow for sharing imaging data.

Choropleth Visualization

4.1 Introduction

Choropleths are simple maps that display regional changes as color (e.g., a US map showing states that lean Democrat vs Republican). Here is an example of a brain choropleth.
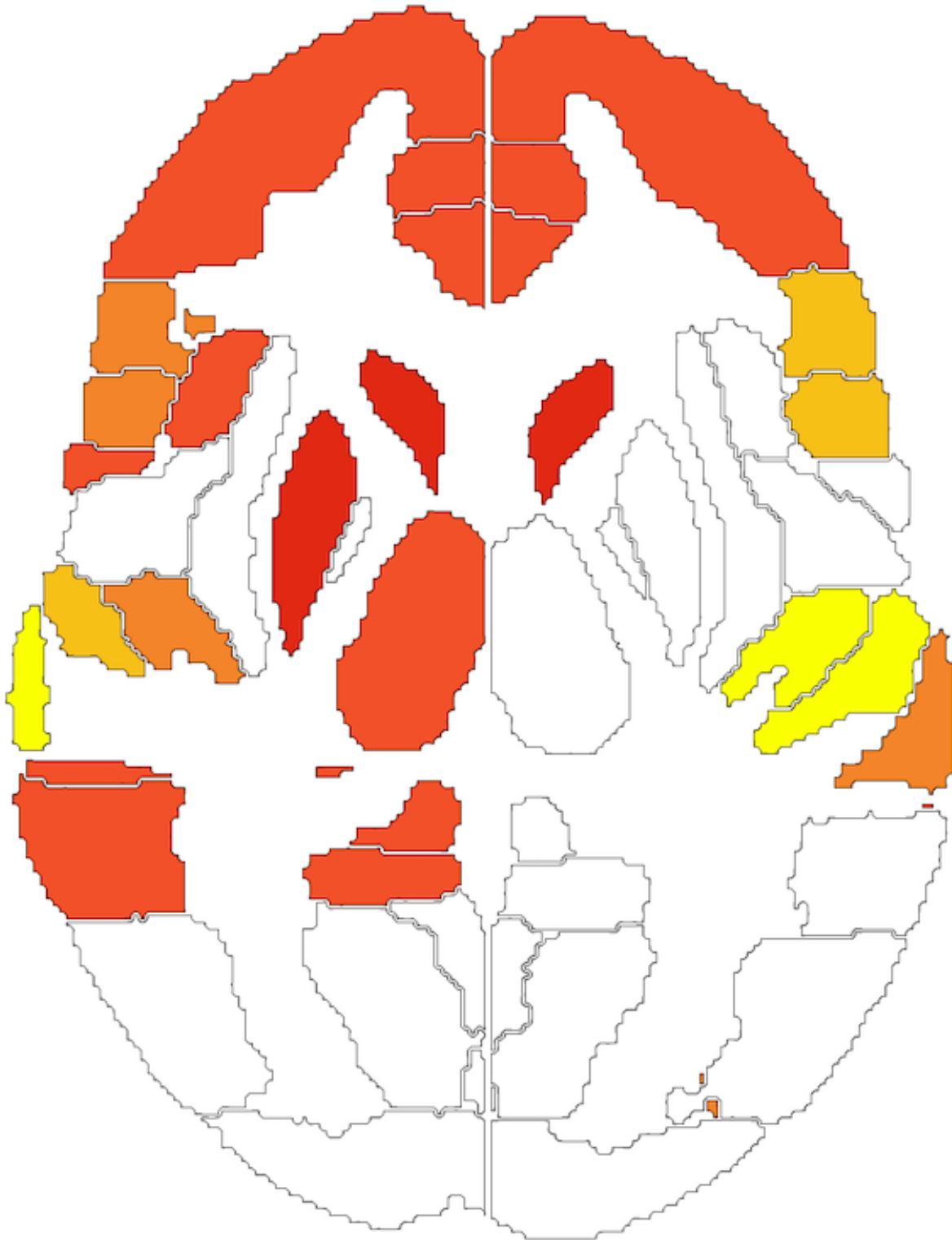
We demonstrated that this simple approach to visualization works well for brains, and, in fact, facilitates analysis:

Patterson, D. K., Hicks, T., Dufilie, A. S., Grinstein, G., & Plante, E. (2015). Dynamic Data Visualization with Weave and Brain Choropleths. PLoS ONE, 10(9), e0139453–17. http://doi.org/10.1371/journal.pone.0139453

Obviously brain choropleths are most valueable if you can choose the slice to view and display information about the regions in each slice. The original dynamic Weave visualizations allowed slice stacking. However, you will need to enable Adobe Flash to view the Weave visualizations, and Weave is no longer available. For this reason, we needed a new implementation. See Devin Bayly's Neuro-choropleths Tool. This work was supported by the University of Arizona HPC center. It uses Javascript to display GeoJSON slices and regional information. Devin's implementation of brain choropleths is fast and easy to use. You can make notes, compare different views and export your session to load up later.

4.2 roixtractor

Even without the brain visualizations, extraction of regional descriptive statistics provides a lot of power for analysis. You can extract such data by region using a Docker container that includes FSL 5.X and the scripts and atlases for extraction. See the roixtractor app provided as part of the scif_fsl tools. The scif_fsl Readme provides more detail. You will need to install Docker to run the tools. The git repository for roixtractor includes all the atlas ROIS, if you are curious.

Note: Docker can be run on some Windows machines, but it requires at least the professional version of Windows 10. You may have trouble with Docker on older operating systems.

Once Docker is installed, you can obtain an up-to-date copy of the scif_fsl Docker container by pulling it from dockerhub:

```
docker pull diannepat/scif_fsl
```

After downloading the Docker container to your local machine, I suggest getting a copy of the wrapper script sfw.sh which facilitates querying and calling the tools in the Docker container. Put this script somewhere in your path. You can begin by running the script with no arguments:

```
sfw.sh
```

Running sfw.sh will provide a list of apps in the container and relevant help for using them. roixtractor is one app in the container and is useful for extracting descriptive statistics for each region in a standard MNI 2mm image. You can try the test statistical image run-01_IC-06_Russian_Unlearnable.nii.

To run roixtractor you must pass it: 1) A statistical image in 2mm MNI standard space 2) A statistical threshold (output regions must be equal to or greater than this value) 3) An available atlas: HO-CB or HCP-MMP1:

- HO-CB: The Harvard-Oxford cortical and subcortical atlases combined with a cerebellar atlas.

- HCP-MMP1: Although it is possible to output these values in HCP-MMP1 space, it is probably not a good idea to use the HCP-MMP1 atlas in volumetric space. It is not likely to be terribly accurate).

Here is an example run:

```
sfw.sh xtract run-01_IC-06_Russian_Unlearnable.nii 1.5 HO-CB
```

This uses the keyword `xtract` to call the correct app. In this example we pass in a NIfTI stats image in 2mm MNI space: run-01_IC-06_Russian_Unlearnable.nii. The threshold value is `1.5`, and the atlas is `HO-CB`.

Given the above command, roixtractor will generate a NIfTI file and two CSVs:

- **NIfTI file** with the mean value displayed in each region that meets criterion. This can be viewed in any NIfTI viewer.
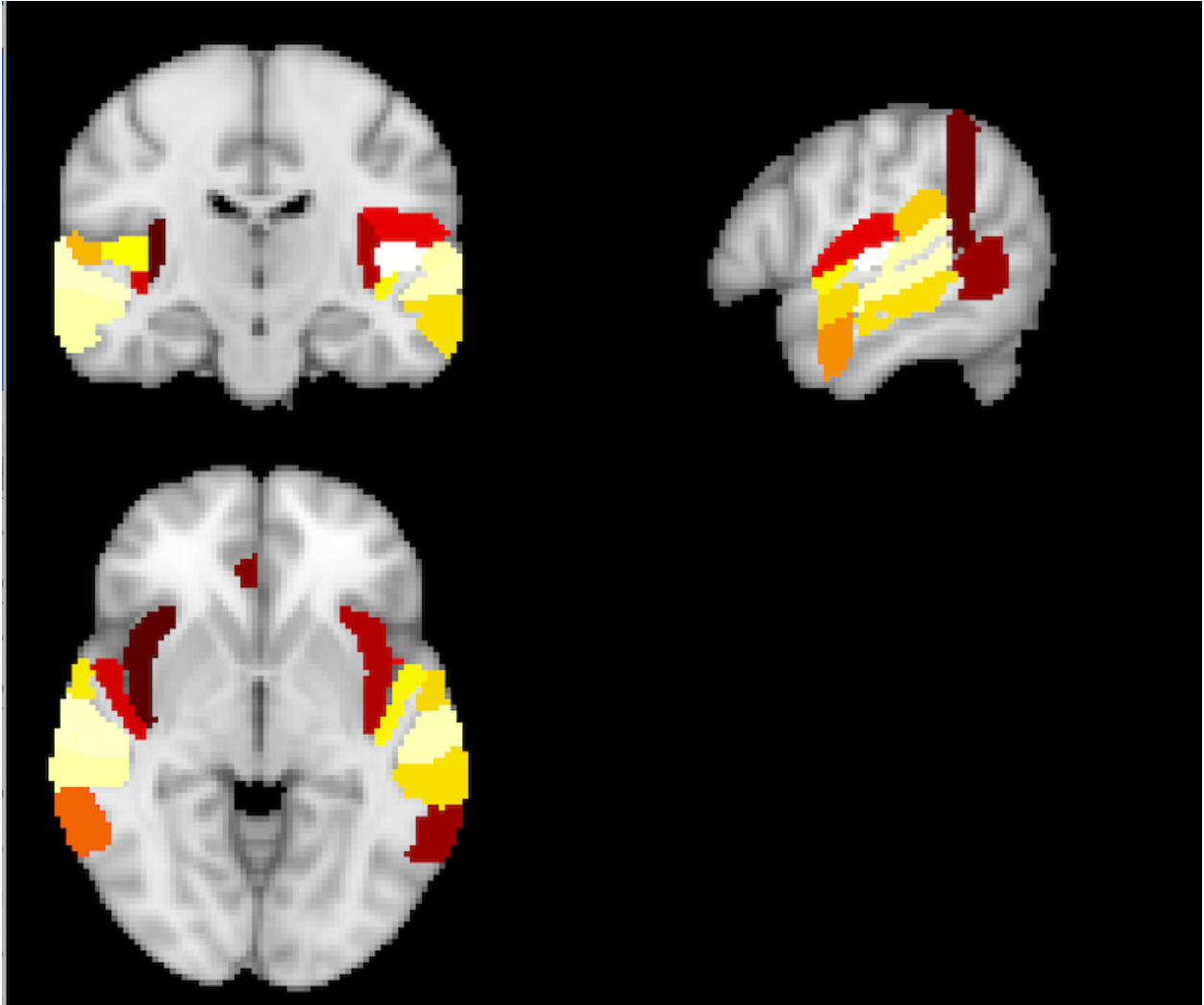
Fig. 1: FSLeyes displays the regions in the hybrid HO-CB atlas that meet the threshold criterion. The image, `HO-CB_run-01_IC-06_Russian_Unlearnable_1.5.nii.gz`, generated by running the above command, is named with the atlas name: `HO-CB` prepended and the threshold value `1.5` appended.

- *The first CSV* (e.g., `HO-CB_run-01_IC-06_Russian_Unlearnable_1.3.csv`), like the NIfTI file, contains only the regions that meet your criterion threshold. This CSV will contain the following fields: *Region, Mean, Max, SD, LR, Lobe, CogX, CogY, CogZ, RegionLongName, RegionVol_MM3, Image*.

- For example: BA44_R,1.664311,2.824052,0.597232,R,Fr,19,71,44, Inferior_Frontal_Gyrus_pars_opercularis_R,5240,run-01_IC-06_Russian_Unlearnable_1.5

1. Region: the short region name

2. Mean, Max and SD are descriptive statistics for each region.

3. LR: identifies the region as left or right (redundant, but useful for sorting)

4. Lobe: The lobe in which that region is found

5. CogX, CogY and CogZ: are the x,y,z center of gravity voxel coordinates for each region

6. RegionLongName: A more detailed name for the region if available

7. RegionVol_MM3: The volume of the region in cubic mm

8. Image: The name of the image you passed to the program: the name of the atlas you used is prepended and the threshold you used is appended to the name. Because image is the last field in the spreadsheet, you can easily use Excel to split the data on underscores. This splitting is useful if you want to concatenate several CSVs from different runs of the xtractor.

- *The second CSV* file (e.g., `HO-CB_run-01_IC-06_Russian_Unlearnable_1.3_LI.csv`) contains laterality information based on Wegrzyn, M., Mertens, M., Bien, C. G., Woermann, F. G., & Labudda, K. (2019). Quantifying the Confidence in fMRI-Based Language Lateralisation Through Laterality Index Deconstruction. Frontiers in Neurology, 10, 1069–15. http://doi.org/10.3389/fneur.2019.00655

The number of Suprathreshold voxels in each region is reported for left and right homologs. Strength=L+R; Laterality=L-R; LI=Laterality/Strength.

4.3 Resources

- Patterson, D. K., Hicks, T., Dufilie, A. S., Grinstein, G., & Plante, E. (2015). Dynamic Data Visualization with Weave and Brain Choropleths. PLoS ONE, 10(9), e0139453–17. http://doi.org/10.1371/journal.pone.0139453: This is the original choropleth paper.

- Neuro-choropleths Iodide Notebook: Devin Bayly, with support from the University of Arizona HPC center, is working on this implementation of a choropleth viewer.

- Sample 3D stats image: run-01_IC-06_Russian_Unlearnable.nii.

- Geo_NIFTI.zip: The regions in NIfTI format for our HO-CB hybrid atlas. This includes ROIS.zip which is individual binary masks for each region and HO-CB_all.nii.gz which combines all the label regions and can be downloaded and used for display of the regions in the hybrid atlas.

- HO-CB GeoJSON files. Only for the curious

- HCP-MMP1 GeoJSON files. Only for the curious

- CreateGeoJSON.zip: Bash and Matlab scripts for creating your own GeoJSON files from NIfTI.

- Tableau_Brains.zip and an accompanying workbook: Tableau_Workshop.zip. A June 2017 implementation of the slices for Tableau. Note, there is no option to scroll through GeoJSON slices in Tableau, you have to pick a static slice for display.

- WeaveTutorial.zip: CSV data suitable for display on the GeoJSON slices (and a complete Weave tutorial) (2015). Deprecated.

4.4 License

The NIfTI format atlas: HO_Cb_all.nii.gz, the 0.5 mm standard space MNI brain: MNI152_T1_0.5mm.nii.gz and the ROIs for all regions are protected by the FSL license Unless explicitly stated, any materials not directly protected by the FSL license fall under the Creative Commons Attribution-NonCommerical 4.0 license

Glossary

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu

Date Created: 2018_08_10

Date Updated: 2019_06_25

Tags: Neuroimaging concepts, software, and educational resources

This is an **alphabetical index of resources and definitions related to neuroimaging**. As such it contains many links. Links and information become stale with time. Please email the maintainer if:

- You encounter a broken link

- You believe some of the information is incorrect

- You'd like to suggest additional entries

Also see the *Bookmarks* page.

Affine transforms Affine transforms are used to register images together. All parts of the image undergo the same transform (i.e., you cannot squish the image in one corner and stretch it in another, you have to stretch or squish everything. An affine transform can be represented as a square matrix of nine numbers. See the interactive affine matrix to get a feel for the transformations.

applytopup Applytopup is an *FSL* tool used to apply distortion and movement corrections to an epi image (DWI and occasionally fMRI). Applytop depends on topup to identify the distortions in images, and both topup and applytopup depend on the acquisition of *reverse phase encode* images during the scan. Applytopup works like this: we have 2 images:

- blip1 consists of two blip up (P-A) B0 volumes. Call these volumes 1 and 2.

- blip2 consists of two blip down (A-P) B0 volumes. Call these volumes 3 and 4.

- applytopup combines the first blip1 volume (1) with the first blip2 volume (3) to create a volume (1+3). Then the 2nd blip1 image (2) is combined with the second blip2 image (4) to create a volume (2+4). These two new volumes are stored in b0_hifi.nii.gz

ASL Arterial Spin Labeling is an MRI modality that characterizes cerebral blood flow. Resources:

- ASL Analysis

- Andy's Brain Book: ASL

BBR In FSL, BBR or *boundary-based registration* is implemented in the script epi_reg, which can use *field maps* to simultaneously register and distortion-correct epi images. See *Difference in Echo times*.

dcm2niix Chris Rorden's excellent dcm2niix converts dicom to nifti. It correctly handles slice timing for multiband images and produces the bids format json sidecar files.

Note that the naming strategy is not necessarily bids compliant. It might be worth giving bids compliant names to your data on the scanner. That should facilitate everything.

If you want to perform slice time correction on multiband fMRI datasets, you should also look at slice timing and multiband and from the bottom of that same page, download stcx.zip

Difference in Echo Times The *difference in echo times* refers to fieldmap characteristics, and is needed for fsl_prepare_fieldmap. The default value for the Siemens is 2.46. To calculate the difference in echo times, you need the TE for the first and 2nd echo: TE=4.92 (first mag); TE=7.38 (phase and 2nd mag); 1 echo (First mag map); 2 echoes (phase map and 2nd mag); 7.38-4.92=2.46. See *BBR*.

Dwell Time Equivalent to *echo spacing* or *time between echoes in k-space*. This is the time it took for the scanner to go from the center of one echo in the EPI sequence to the center of the next.

The echo spacing should be listed in the parameters of the scan (e.g., 0.94 ms for my 32 direction B=1000 dwi scans). Effective echo spacing= *echo spacing* divided by *grappa* (a.k.a. acceleration, ParallelReductionFactor-InPlane) factor (2 in this case) and then divided by 1000 to get units in seconds instead of ms.

e.g., (0.94/2)/1000=0.00047

Effective Echo Spacing is sometimes reported directly in the BIDS json file, e.g., in sub-001_acq-AP_dwi.json: "EffectiveEchoSpacing": 0.00047001

Echo Spacing See *Dwell time*.

Effective Echo Spacing See *Dwell time*. To run epi_reg with field maps, we need the effective echo spacing of the epi image being processed.

Field Maps Field maps can be used to correct distortions in epi images (like dwi and fmri). A set of field maps contains 3 images: two magnitude images and a phase image. See *BBR*. They take about 1 minute of scan time to acquire. FSL has some nice tools for handling Siemens field maps. You may also wish to read a more general description of field maps.

FreeSurfer FreeSurfer is a unix-based (mac or linux) neuroimaging workbench with tools for fMRI, dwi and especially structural segmentation and cortical thickness analysis. Andrew Jahn provides a useful sequence of video lectures about FreeSurfer.

FSL FSL is a unix-based (madc or linux) neuroimaging workbench with tools for fMRI, structural and especially dwi processing. FSL stands for *FMRIB Software Library*.

FSLeyes FSLeyes is a unix-based image viewer for *FSL*. It is Python-based and extensible. Andrew Jahn provides a useful sequence of video lectures about FSLeyes.

GIFT GIFT is a sophisticated Matlab toolbox for running ICA analysis. Additional materials are available here. These include a slightly different user manual (MAC oriented and exploring some different topics). The manual includes links to panopto videos that walk you through the analysis. Sample data, a 6 minute walkthrough video, presentations and sample batch files are also available.

grappa GeneRalized Autocalibrating Partial Parallel Acquisition (a.k.a acceleration or ParallelReductionFactorIn-Plane) is a multicoil parallel imaging technique. For more information, see the excellent MRIQuestions or MR-tip.

NIFTI Headers and Orientation Imagine you have created a mask image, but it is not in quite the right place relative to the anatomical underlay. Where is the mask? We can characterize it in terms of voxel coordinates OR in terms of left, right, anterior posterior etc. This is *voxel space* and *world space* respectively. Just because a display tool (like FSLeyes) allows you to nudge the position of the mask, does not guarantee that the underlying voxel positions are changed. This is very confusing: when you look at the mask in the display tool, it is displayed in the new nudged position, but when you perform image math operations, it is as if the position had never been nudged.

In FSLeyes 0.27.3 you need to save the affine transformation corresponding to your nudge and then apply it to the mask with FLIRT if you want to actually change the baked in position of the mask. This is similar to coregistration and reslicing which actually change the baked-in positions of the intensities at each voxel. Once you have changed the actual values of the voxels, your mask will not only display in the nudged position but will actually BE in the nudged position and thus useful for all image math tools. See also FSL's Orientation Explained.

There is another wrinkle. There has to be some way to relate voxel space to world space. The NIFTI header contains two (not one, sigh) ways to encode this relationship: the sform and qform. S seems to stand for *standard* whereas Q stands for quaternion. This has led to two different camps: the sform aficionados (SPM, FSLeyes, Mango, MRIcroGL) and the qform aficionados (ANTS, ITK-SNAP, MRtrix). The sform aficionados record any change in the world-to-voxel relationship in the sform matrix; however the qform aficionados record it in the qform. Then each camp has to make a choice about what to do with the other camp's matrix:

1. Leave the other matrix alone, allowing the two matrices to be different.

2. Set the other matrix the same as your matrix, so any software will see the change.

3. Set the other matrix to all zeros (and assume that *smart* software will look at the other matrix when their favorite matrix has been set to zeros.

As different versions of a particular piece of software have arrived, developers have sometimes switched strategy. For example, SPM used to update only the sform (strategy 1) but more recently has adopted strategy 2. ITK-SNAP uses strategy 3. MRtrix generally uses the qform now, but you can set a preference to use the sform instead (which used to be the default). If the sform and qform differ, MRtrix will warn you.

Phase Encode Direction If you are using *field maps*, epi_reg requires a phase encode direction (–pedir) for the primary dwi image. Unfortunately, the BIDS JSON files use i,j,k and *FSL* uses x,y,z to record this value. "PhaseEncodingDirection": "j-" in the json file corresponds to -y in FSL's terminology (-y=AP).

Q-form A part of the NIFTI header. See *NIFTI Headers and Orientation*

Q-space Sampling For DWI, the distribution of gradients used to sample the diffusion space. An interactive tool for exploring q-space sampling (and creating your own gradient tables): q-space

Reverse Phase Encode DWI data are acquired in one direction (e.g., A->P). Because the echo time for DWI images is so long, this means there is tremendous distortion. If you look at A->P images in axial view, you'll see stretchy eyeballs. A second image (1-2 B0s) in the opposite phase-encode (P->A) direction takes ~45 seconds to acquire and can be used to help correct distortion in the first image. This second image experiences tremendous distortion in the opposite direction of the original A->P images. If you look at the P->A image in axial view, you'll see it has squishy eyeballs.

- The default Siemens A-P: stretchy eyeballs, (negative phase encode, blip down -1)

- The reverse phase encode B0 P-A: squishy eyeballs, (positive phase encode, blip_up 1)

- In *FSL* we can use topup to calculate the distortion parameters based on both images. We can then use applytopup to apply these parameters to one of the images. This can be helpful for creating a corrected B0 image for skull stripping.

- N.B. My experiments suggest it is better to correct the A-P data (less noise, fewer spikes and holes in weird places, better eyeball reconstruction).

- Finally, we feed the topup results into eddy for correcting eddy currents and movement.

- The following text files contain parameters needed for processing: * acqp_A-P * acqparams.txt * index_A-P.txt

- The acqp file is used by eddy and topup. This is a text-file describing the acquisition parameters for the dwi volumes. In eddy this parameter file is passed with the flag –imain. In topup the parameter is passed with the flag –datain.

- The value is repeated 32 times for the 32 volumes. The -1 indicated the phase encode direction for y in A-P. The last value is *Total Readout Time*.

- acqparams.txt: Used for topup. Specifies the values for the 4 blip_down and blip_up B0 volumes

- index_A-P.txt: Used by eddy to index all the values in acqp_A-P.txt (all 1's)

S-form A part of the NIFTI header. See *NIFTI Headers and Orientation*

Topup Topup is an *FSL* tool primarily used together with applytopup and eddy to correct distortion in DWI (and occasionally fMRI) images. Topup relies on *reverse phase encode* images.

Total Readout Time Total readout time is the echo spacing (*dwell time*) * the number of phase encoding steps. Total readout time applies to the epi sequence (usually dwi) and appears in the acqp and acqparams files used by *FSL*'s topup. If you use grappa, divide the number of phase encoding steps by the grappa factor. For example, if echo time=0.94 ms (but we want it in seconds, so 0.00094 sec), phase encoding steps=128, and grappa=2 then total readout time is (128/2)*.00094 sec =0.06016 sec

Wavelets An accessible introduction to wavelets

Image Processing Tips and Tricks

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu

Date Created: 2019_08_17

Date Updated: 2019_09_14

Tags: anat, segmentation, image formats, header editing, registration

Most of the tools described below are based on FSL. FSL is not the only tool that can be used for this sort of processing, it is just the tool I know best. Where appropriate, bash scripts are available for download from my Bitbucket tools site. You can just download the whole repository if you want, but it'll include scripts that may be of no interest. Otherwise, the scripts you want are described in each section (left click a link to view the script, right-click to download). Also, see the *Clinical Scans* page for issues related to processing data from clinical scans (which may be CTs and or have very low resolution in two of the three planes).

6.1 Anatomical Image Preparation

Strutural images, usually T1-weighted, but sometimes T2-weighted, FLAIR, scalar DWIs (B0, FA, MD etc.) or even CTs may be available as underlays to draw lesion masks on. CTs are a really special case and are discussed separately on the *Clinical Scans* page. T1w images can be processed with a single script described in *Prepare T1w Image* that reorients, bias-corrects, crops and even generates a brain mask. Several of these processing steps can also be handled separately as described in other sections below.

6.1.1 Prepare T1w Image

For a T1w image, you can use prep_T1w.sh to run fslreorient2std, robustfov, bias correction, and to create a brain mask for a T1w image. Note that prep_T1w.sh depends on FSL and on two additional scripts being present: fsl_anat_alt.sh and optiBET.sh:

```
prep_T1w.sh sub-001_T1w
```

Optionally, if you have a lesion mask in native space, you can ensure that it also gets cropped and reoriented by adding it to the `prep_T1w.sh` command:

```
prep_T1w.sh sub-001_T1w lesion
```

Check the brain mask! Large lesions, especially chronic lesions near the surface of the cortex, frequently result in a brain mask that does not have enough coverage. If all of your data consists of high quality research scans, then consider using *SPM12 normalization* before you delineate the lesion. That way, you can use the standard brain mask instead of generating a brain mask.

Running `prep_T1w.sh` creates a subdirectory named after your anatomical file, e.g., `sub-001_T1w.anat`. It also backs up your original files, e.g., `sub-001_T1w_orig.nii.gz` and optionally your lesion file if you processed it, e.g., `lesion_mask_orig.nii.gz`. The suffix `_orig` has been added to these original files. Your new cropped images will have the names that you originally passed in (e.g., `sub-001_T1w.nii.gz` and `lesion_mask.nii.gz`).

6.1.2 fslreorient2std

It never hurts to reorient the image to standard orientaion. Some viewer programs will show you the image in standard orientaion whether or not it is stored that way. Other image viewers will show you the image as it is stored. For the sake of consistency, ensure the image is stored in the same orientation as the MNI standard. You can try this test data to see fslreorient2std in action: sub-001_T1w.nii.gz (this is my head, so no worries about HIPPA, it is free to use):

```
fslreorient2std sub-001_T1w sub-001_T1w_reoriented
```

The command is harmless if the data is already correctly oriented. If you apply the above command to the ITK-SNAP tutorial data T1w_defaced.nii.gz, nothing changes.

6.1.3 Crop Neck with robustfov

Crop the anatomical images, thus removing the lower head and neck. This command will run fslreoreint2std if it has not already been done.:

```
robustfov -i sub-001_T1w_reoriented -r sub-001_T1w_cropped
```

6.1.4 Correct Extreme Anisotropy

Clinical images often have extremely anisotropic voxels. This is not well handled by tools like MRIcron. In addition, ITK-SNAP needs isotropic (or near isotropic) data to do good semi-automated segmentation. You can easily make voxels isotropic with flirt: A simple FSL-based script to facilitate creating isotropic images (anatomical images or masks) is available here.: iso.sh. To try the script out, download T1w_aniso.nii.gz:

```
iso.sh T1w_aniso 1
```

The above command takes `T1w_aniso.nii.gz` as input and reslices it to 1mm isotropic voxels. The output is called `T1w_aniso_1mm.nii.gz`. The output image can now be used in ITK-SNAP or MRIcron. Note that the image is just as blurry, but it has more slices in the z direction now. That means any mask you draw will be smoother.

6.1.5 Reslice Other Anatomicals into One Space

You can use the script reslice.sh to facilitate running FSL flirt:

```
reslice.sh anat_FLAIR anat_T1w
```

- This example reslices an image, `anat_FLAIR.nii.gz` to be in the same space as `anat_T1w.nii.gz`.

- The script can be used, for example, to ensure that images of several different modalities are in the same space so they can be used with *3D segmentation clustering in ITK-SNAP*.

- **Clinical CT** Another useful application of reslice.sh is to apply *gantry tilt correction* (Use the gantry tilt corrected image as the reference image).

- **Fusion Images** In the section on *fusion images*, I walk you through using reslice.sh to create a very simple fusion image.

> **Warning:** reslice.sh uses trilinear interpolation, which is fast and fine for anatomical images. However, it is NOT appropriate for reslicing masks.

6.2 Masks

A mask (such as a lesion mask) should be binary, that is, it should contain ONLY 0's and 1's. This is because statistical programs for evaluating lesion masks, along with various image math procedures often assume the values are 0's and 1's. So, we want to make sure the masks remain binary.

Some tools, like MRIcron, offer a smoothing option for masks. The problem is that smoothing may be accomplished by softening the edges which means that those values are interpolated (i.e., 1's are replaced with values like 0.5, 0.3 etc). In addition, if you register your mask to a different image space you can introduce interpolated values if you fail to use *Nearest Neighbour* interpolation. You should go through the *Hidden Problems Section* below, before registering a mask. In this section, I offer several tools for improving masks without compromising their binary nature.

Ensure Masks are Binary

If you reslice a mask into a different space, you should do so with *Nearest Neighbour* interpolation. Otherwise, values at the edges of the mask may be interpolated between 1 and 0. Smoothing a mask may result in the same problem. To ensure masks are binary, use fslmaths:

```
fslmaths lesion -bin lesion -odt char
```

Check that a mask contains correct values (`-M` Mean of non-zero voxels should be 1. `-S` Standard deviation of non-zero voxels should be 0:

```
fslstats lesion -M -S
1.000000 0.000000
```

6.2.1 Erode

Remove islands and spikes by eroding the edges of the lesion mask. erode.sh is a bit less complex to call than the fslmaths command. In the command below, we do one iteration of the erosion and do it in 3D (default is 2D):

```
erode.sh lesion 1 3D
```

> **Warning:** Be Careful with erosion! If you run too many iterations on a small mask, the mask will disappear! You might want to make a backup copy of your mask first.

6.2.2 Dilate

Fill tiny holes in the mask by dilating the edges of the lesion mask. dilate.sh is a bit less complex to call than the fslmaths command. In the command below, we do one iteration of the dilation and do it in 3D (default is 2D):

```
dilate.sh lesion 1 3D
```

6.2.3 Fill Holes

fslmaths has its own hole-filling algorithm. Simply add the appropriate flag to fslmaths. | -fillh : fill holes in a binary mask (holes are internal - i.e. do not touch the edge of the FOV) | -fillh26 : fill holes using 26 connectivity | In the command below we input the lesion_mask, fill the holes and output it with the same name. Finally, we ensure it is the correct datatype:

```
fslmaths lesion -fillh lesion -odt char
```

6.2.4 Cluster Count

It can be difficult to determine whether a lesion mask is one big volume or whether little islands have broken off. cluster_count.sh tells you how many separate clusters are in the mask, how big they are and the center of each one. In general, I can't imagine why you'd want to keep clusters of less than about 10 voxels. In the command below we ask how many clusters are in lesion_mask.nii.gz and we set connectivity to 6.

Connectivity can be set at 6, 18 or 26, but always defaults to 26.

- If the connectivity is set to 6, then voxels only belong to a single cluster if their faces are in contact.

- If the connectivity is set to 18, then voxels belong to a single cluster if their faces OR edges are in contact.

- And, if the connectivity is set to 26 (the default if you don't specify a number), then voxels belong to a single cluster if their faces, edges or corners are in contact.

So, setting the connectivity to 6 ensures the greatest possible number of clusters will be counted:

```
cluster_count.sh lesion 6
```

6.2.5 Cluster Clean

If there are several clusters, especially if there are several tiny clusters, you may wish to remove the smallest ones. cluster_clean.sh can remove islands without the dangers of erosion. Often you'll find you have one gigantic cluster (many thousands of voxels) and several clusters of 1-10 voxels. You can safely remove the tiny ones with the cluster_clean.sh. A command like the following which removes any clusters smaller than 10 voxels from the mask image:

```
cluster_clean.sh lesion 10
```

As with cluster_count.sh, a third argument can be added to specify connectivity of 6 or 18. The default connectivity is 26:

```
cluster_clean.sh lesion 10 18
```

6.2.6 Smooth

Yes, this is what I warned you against, but with some care it seems to work in FSL and produce binary results. You may have to play with the `-kernel gauss 0.1` setting a bit. This seems to do very gentle smoothing, but still dilates the mask (and larger kernel values are more extreme). One iteration of erosion afterwards seems to do the trick. I ran this smoothing three times, each time running on the output of the previous iteration, and then eroding the results. The effects were improved:

```
fslmaths lesion -kernel gauss 0.1 -fmean -bin lesion_smooth
erode.sh lesion_smooth 1 3D
```

6.2.7 Left-Right Flip and Nudge

It can be useful to flip the normal ventricles in one hemisphere onto the abnormal ventricles in the other hemisphere to characterize ventricular enlargement. This is a two step process:

See *Left-Right Flip* and *Nudging*

6.3 Check for Hidden Problems

The section above deals with ways to clean up the way a mask looks, ensuring that it is shaped organically, like the lesion, and is not filled with simple drawing errors. However, before using a mask for image math or statistical analyses, you want to make sure that more insidious problems are not present. The checks below can prevent some of these more insidious problems for both masks and antomical images.

6.3.1 fslinfo and fslhd

fslinfo provides very limited information about your images. fslhd provides more. Both are useful. Try them like this:

```
fslinfo lesion
fslhd lesion
```

6.3.2 Ensure Zeros, not NaNs

SPM uses NaN (Not-a-Number) in addition to zeros in NIFTI images. However, some processing programs (e.g., FSL) do not handle NaN well at all. In fact fslstats will add NaN values in a mask image to the ones in that mask image, and FSLeyes will display both NaNs and ones as values distinct from zero (right-hand image below) if the range is set from -1 to 1. The bottom line is that it is a better and more robust choice for your mask images to use only zeros and ones. The following FSL command will convert all NaN values in `lesion` to zeros in `lesion2`. The resulting 1's and 0's in lesion2.nii.gz will be handled correctly by all viewing programs:

```
fslmaths lesion -nan lesion2
```
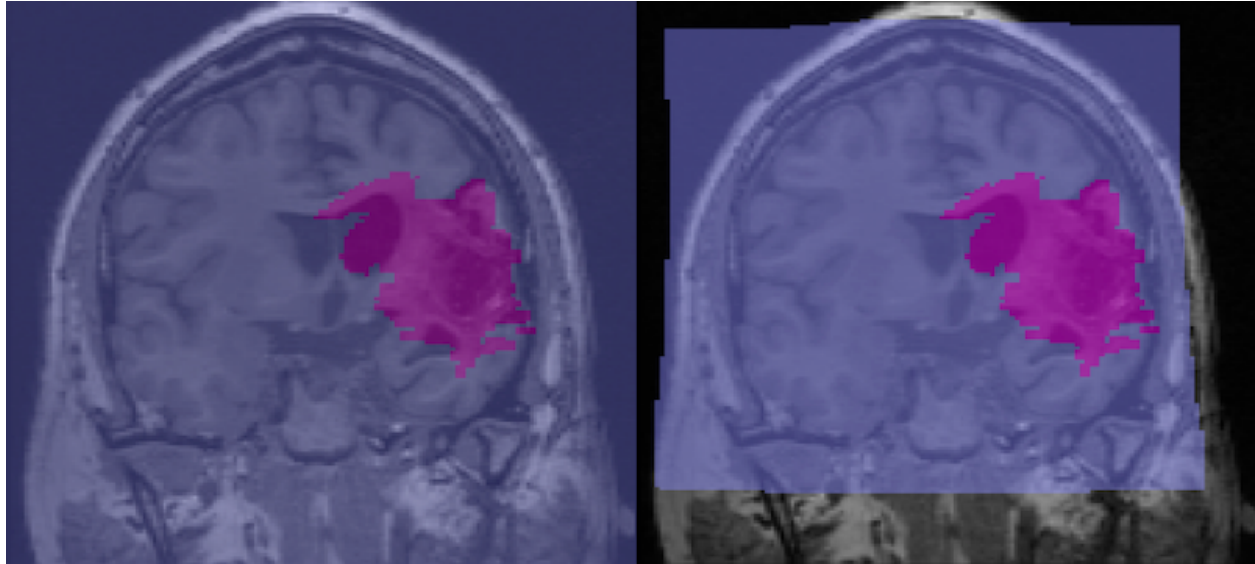
6.3.3 Ensure SFORM and QFORM Match

The image header for NIFTI has two separate ways to represent positional information in a viewer. Some programs (ITK-SNAP) prefer to edit the qform and others prefer to edit the sform (SPM, FSL), but these choices have evolved over the years along with what happens to the other matrix when the preferred one gets edited (set the other to 0s, match the other to the preferred, leave the other as it was). ITK-SNAP sets the sform to 0s, a potential problem for other viewers. The following tools can help: sqdiff.sh tells you whether the sform and qform of an image differ. sqdiff_wrap.sh will run sqdiff on every image it finds in subject subdirectories (sub*) of the current directory. q2s.sh copies the qform to the sform. s2q.sh copies the sform to the qform. All are based on the fsl tool *fslorient*.

Note: Because the sform and qform are stored differently, they have different levels of precision. It is possible to copy the sform to the qform and still find that they are reported to be a bit different by sqdiff.sh. If you copy the qform to the sform, however, any differences will be completely removed.

If you created a lesion_mask.nii.gz in ITK-SNAP, then your header will contain a qform but not sform:

```
sqdiff.sh lesion

diff results for Temp lesion_mask.nii.gz:
1c1
< 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
---
> -0.999994 -0.00314216 0.00112513 96.8904 -0.00315573 0.99992 -0.0122661 -90.1013 0.
↪0010865 0.0122696 0.999924 -77.9563 0 0 0 1
```

sqdiff.sh compares the sform and qform. You see results if they are different. In the example above, ITK-SNAP has set the sform to `0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1` which is quite different that the qform!

You can also call sqdiff_wrap.sh to get a report on all sform and qform data in sub* subdirectories of the current directory:

```
sqdiff_wrap.sh >> sqdiff.txt
```

To make the sform and qform match (a very good idea), you can run either q2s.sh or s2q.sh. After generating a mask with ITK-SNAP, you can see that the sform has been zeroed out, so we want to copy the qform to the sform:

```
q2s.sh lesion
```

Both q2s.sh and s2q.sh run on specified image file argument (as illustrated above). However, if you type the script name with no arguments, you get a choice to run on all images in the current directory or to view the help message.

6.3.4 Ensure Standard Orientation in NIFTI Header

Standard orientations should be `Right-to-Left` for sform and qform orient (e.g.)

qform_xorient Right-to-Left
qform_yorient Posterior-to-Anterior
qform_zorient Inferior-to-Superior

If you encounter an image with the following:

qform_xorient Left-to-Right
qform_yorient Posterior-to-Anterior
qform_zorient Inferior-to-Superior

then some programs will have trouble (iMango and Papaya will get the L-R wrong as of May 23, 2019). Note that this is not proper behavior for such programs, and it should be reported as a bug (it has been) Nevertheless, it doesn't hurt to make sure everything is consistent:

See get_orient.sh to view orientations. If you have run fslreorient2std on sub-001_T1w_reoriented.nii.gz, you'll find it is still in Left-to-Right orientation:

```
get_orient.sh sub-001_T1w_reoriented
```

See reset_orient.sh to switch the orientations and then view the results:

```
reset_orient.sh sub-001_T1w
```

6.3.5 Save Space on your Hard Drive

FSL has the annoying habit of saving images with a high bitdepth (32 bits). You don't need that for structural T1w images (or T2w, or FLAIR). It justs wastes a lot of space. DWI images often use floating point values and should remaim 32 bit.

Ensure anatomical images are short (16 bit):

```
fslmaths anat_T1w anat_T1w -odt short
```

Ensure masks are binary and char (8 bit):

```
fslmaths lesion -bin lesion -odt char
```

6.3.6 Create a Brain Mask

It can be very useful to have a brain mask to trim the lesion if the filling algorithm leaks outside the brain. There are many ways to generate such a mask: BET in FSL , optiBET, and MRIcron.

If you are working with T1w_defaced.nii.gz, you may wish to use the native_brain_mask.nii.gz that I have generated with optiBET.

If you have normalized the head with SPM12 following *our suggested approach*, then you can use this standard brain mask MNI_T1_1mm_brain_mask.nii.gz. This is the 1mm MNI brain mask cropped into the space of the final SPM12 output.

If you generate your own brain mask, ensure that it has good coverage. After you have generated a lesion mask with ITK-SNAP, you should check that the lesion has not leaked outside of the brain mask (yellow arrow in figure below).
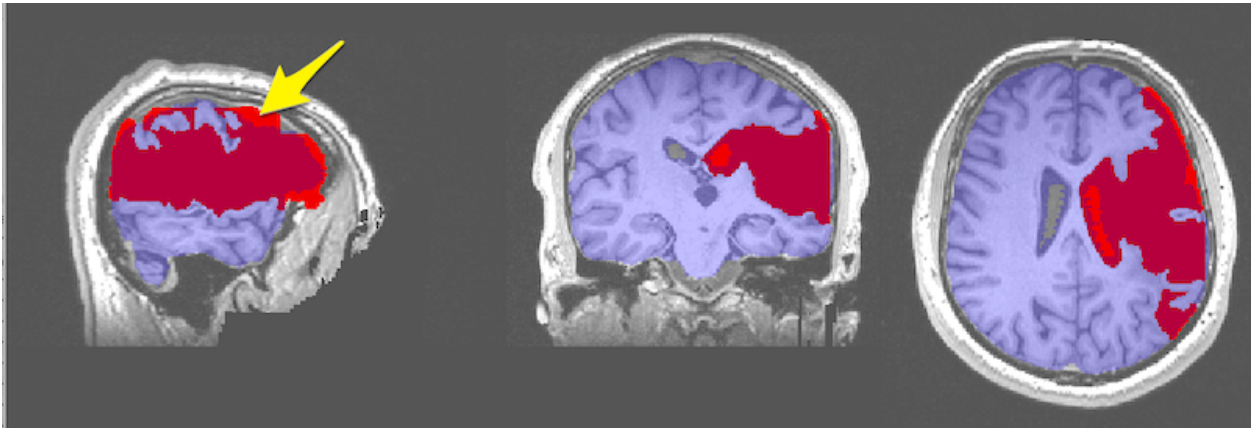
You can look at the T1w image, the brain mask and the lesion mask with FSLeyes. There are several ways to start FSLeyes, but the command below shows you some of the options you can pick at the command line:

```
fsleyes T1w_defaced -b 70 -c 70 lesion_mask -cm Red native_brain_mask -cm Blue -a 30
```

This command says: Use fsleyes to view the T1w_defaced.nii.gz image. FSL looks for files with the `.nii.gz` suffix, so supplying that at the command line is optional. Because the image is usually too dark on my monitor, I increase the brightness `-b` from the default 50% to 70%. I do the same with contrast `-c`. Next, I want to add the lesion_mask with the red color mask `-cm Red`. Next, I want to add the native_brain_mask with a blue color mask `-cm Blue` and with alpha transparency of 30% `-a 30`.

You can use a command like this to trim the lesion so it only occupies the region that overlaps the brain mask, e.g.:

```
fslmaths lesion_mask -mul native_brain_mask -bin lesion_mask -odt char
```

This command says: Multiply the lesion_mask and the native_brain_mask, retaining only voxels that are in both. Then binarize the lesion mask with `-bin` and ensure that it doesn't take up too much space on your drive with `-odt char` which sets the output datatype to 8 bits.

Note: Especially in native space, the brain mask may have inadequate coverage. This is partly because brain masking is a hard problem, and partly because brain masking is a REALLY hard problem when there is a large lesion near the surface of the brain. Scroll through the 3 orthogonal slices and think carefully about whether you want to trim the lesion with the brain mask.

6.3.7 Left-Right Flip Lateral Ventricle Mask

Many of the large lesions we mask are contiguous with the ventricle and result in abnormal enlargement of that ventricle. To estimate the abnormal enlargement, one can segment the normal ventricle in the other hemisphere (assume the right ventricle for this discussion) and then Left-Right flip the segmentation. In standard space, this will generally be aligned in the opposite hemisphere, but in native space, where the head is often not perfectly straight in the scanner, it may be necessary to nudge the flipped segmentation by eye. Here I describe a combination of ITK-SNAP and FSL tools and some related concerns:

As described above for lesion masking, segment and save the right lateral ventricle. Call it LV_R.nii.gz. Left-Right flip the segmentation using FSL command line tools. fslswapdim takes the mask `LV_R.nii.gz` as input and flips on the x axis (which is a left-right flip):

```
fslswapdim LV_R -x y z LV_L
```

> **Warning:** The L-R flip works in isolation. That is, if you flip x only `-x y z` it works fine, but if you try to flip other orientations at the same time, e.g. `-x -y z` then x won't flip. Weird, so keep it in mind. See Orientation Explained

In conjunction with flipping, it may be necessary to use nudge.sh (described below in *Nudging*), especially if you are working in native space. This is because the native space images are often tilted, so the two hemispheres may not be nicely aligned on a given slice.

6.3.8 Nudging

Tools like FSLeyes and MRIcron may allow you to move a mask image around relative to the anatomical underlay. But results may not carry over to image math operations or other pieces of software (Many thanks to Paul McCarthy of the FSL group for explaining this and providing an example of the right flags to provide to flirt). See the description of the problem in *NIFTI Headers and Orientation* and in the section on *Sform and Qform*.

Download and call nudge.sh to open fsleyes with the anatomical underlay and the mask you want to nudge:

```
nudge.sh T1w_defaced LV_L
```
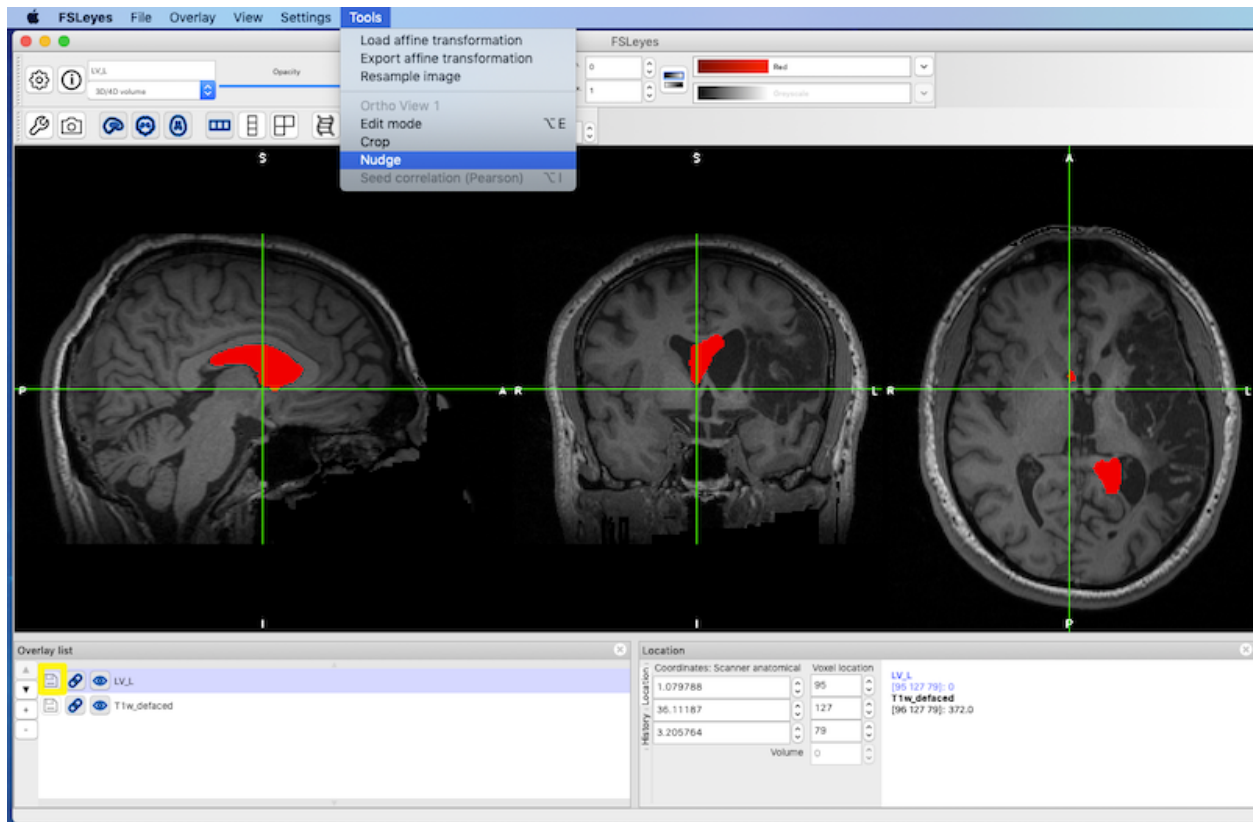
- Nudge the ventricle mask over onto the left (red box, figure below).

- When you save the nudged mask, choose *Apply* (green box, figure below) and then *Close* (blue box, figure below).

- Save the changes by clicking the floppy disk icon (yellow box, bottom left of figure below) and choose to overwrite the existing mask.

- Finally, close FSLeyes. nudge.sh will apply the transformation to your mask.
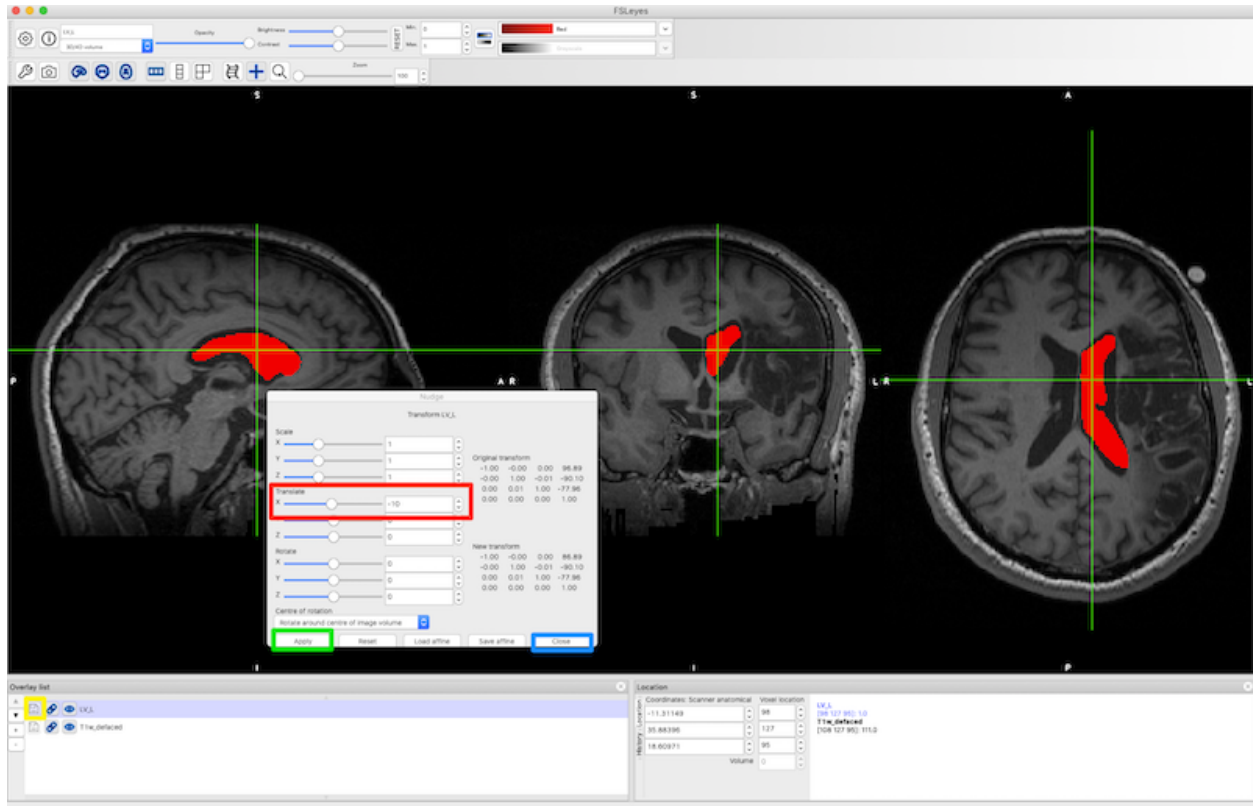
Note: FSLeyes 0.27.3 adjusts the sform/qform matrices in the NIFTI header to project different images into a common display coordinate system. However, saving the nudged mask in FSLeyes does not currently make the image suitable for use with tools like fslmaths which require that the sform/qform of the images match (Personal Communication, 3/28/2019, Paul McCarthy). The nudge.sh script opens FSLeyes with the relevant images and then applies the necessary flirt transform to ensure that the nudged mask is ready for use with fslmaths. At some future point, FSLeyes will make the transform when you save the image, and the nudge.sh script will be irrelevant.

Rotation

Nudge allows translations and rotations. Rotations are tricky. Consider, for example, a Sylvian Fissure mask flipped from the other side of the brain. I suggest these strategies:

1. Use the sagittal view, so you can see the relative position of the mask and the superior edge of the temporal lobe. You want your mask to sit directly above the temporal lobe and be aligned with it.

2. When you choose Nudge from the FSLeyes Tools menu, you have 2 choices for rotation: `Rotate around centre of image volume` or `Rotate around current cursor location`. For this task, `Rotate around current cursor location` is the more intuitive choice. Just make sure you locate your cursor at the center of the long axis of the mask.

3. To rock the mask in the sagittal view, rotate around the x-axis.

4. Remember there is a reset button

6.3.9 Jaccard

Once you have nudged a mask, you want to make sure that it has moved in both world and voxel space. nudge.sh saves a `*.back` image (e.g., LV_L_back.nii.gz) which you can use to compare. Remember, looking at the images in a viewer is not enough to gaurantee the nudge has been applied. You can use this simple jaccard.sh script that tells you how much two masks overlap. In the first case, `jaccard.sh` warns us that the images are not in the same space and that it will compare the two masks using voxel-based orientation. It then reports that Jaccard Index is 1 (the overlap is complete) and Jaccard distance is `0` (there is no distance between the masks):

```
>jaccard.sh cube cubeW

WARNING:: Inconsistent orientations for individual images in pipeline!
         Will use voxel-based orientation which is probably incorrect - *PLEASE␣
↪CHECK*!

Total voxels cube is 125000
Total voxels cubeW is 125000
Intersect voxels are 125000
Union voxels are 125000
Jaccard index is 1.000000
Jaccard distance is 0
```

In this second case, the images are in the same space (there is no warning about inconsistent voxels). In addition, the images do NOT overlap completely (Jaccard Index is less than 1; Jaccard distance is more than `0`):

```
>jaccard.sh cube cubeW+Vox
Total voxels cube is 125000
Total voxels cubeW+Vox is 125000
Intersect voxels are 61250
Union voxels are 188750
Jaccard index is .324503
Jaccard distance is .675497
```

6.3.10 Image Math

Once you have nudged the ventricle mask into the correct location on the left, you can subtract it from the lesion mask, again, using FSL command line tools:

```
fslmaths lesion.nii.gz -sub LV_L.nii.gz -bin lesion_rev.nii.gz
```

You will need to examine the results and clean up slim isthmuses left over by the subtraction. A benefit of this approach is that you have the mask used to remove the normal ventricle, so you can always repeat the process exactly.

Several useful tools for cleaning up masks are described here: *Useful Image Processing Tips and Tricks*. In addition, there are details about what is stored in the NIFTI image headers and problems that can arise when working with these data, especially across multiple image processing tools.

6.4 Other

6.4.1 Edit Image Header (Mango)

If you encounter images with incorrect dimensions, you can revise the header using *Mango Header Editing*. This is handy if you have an image with the wrong slice thickness, for example.

6.4.2 Visualization of Lesions using Maximum Intensity Projection

1. In FSLeyes, load and display the anatomical image and two copies of the lesion file.

2. By default the lesion file is displayed as a *3D/4D volume* which displays the lesion mask for each slice. In the figure below, the 3D lesion is displayed in red (and has a red box around it in the *Overlay list* on the bottom left of the figure).

3. Set the second lesion file to display as a *Max Intensity Projection* (MIP). It needs to be underneath the lesion file above, as it will probably be bigger. You can find the *Max Intensity Projection* in the upper left corner and the corresponding copy of the lesion file in the *Overlay list* (pink boxes, figure below). The MIP is also displayed in pink.

4. Settings for the lesion maps are available from the gear on the upper left of FSLeyes (yellow box, figure below) and include an option to invert the colormap and otherwise refine the display. To get this particular shade of pink, we chose the pink color map, then *Invert colour map* (green box, figure below) and we set the *Display range Max* to 2. It also seems to work to use the default pink color map and set the max to about 4.

5. We chose the ch2.nii.gz standard image, and set the x coordinates to -35 (orange box, figure below).
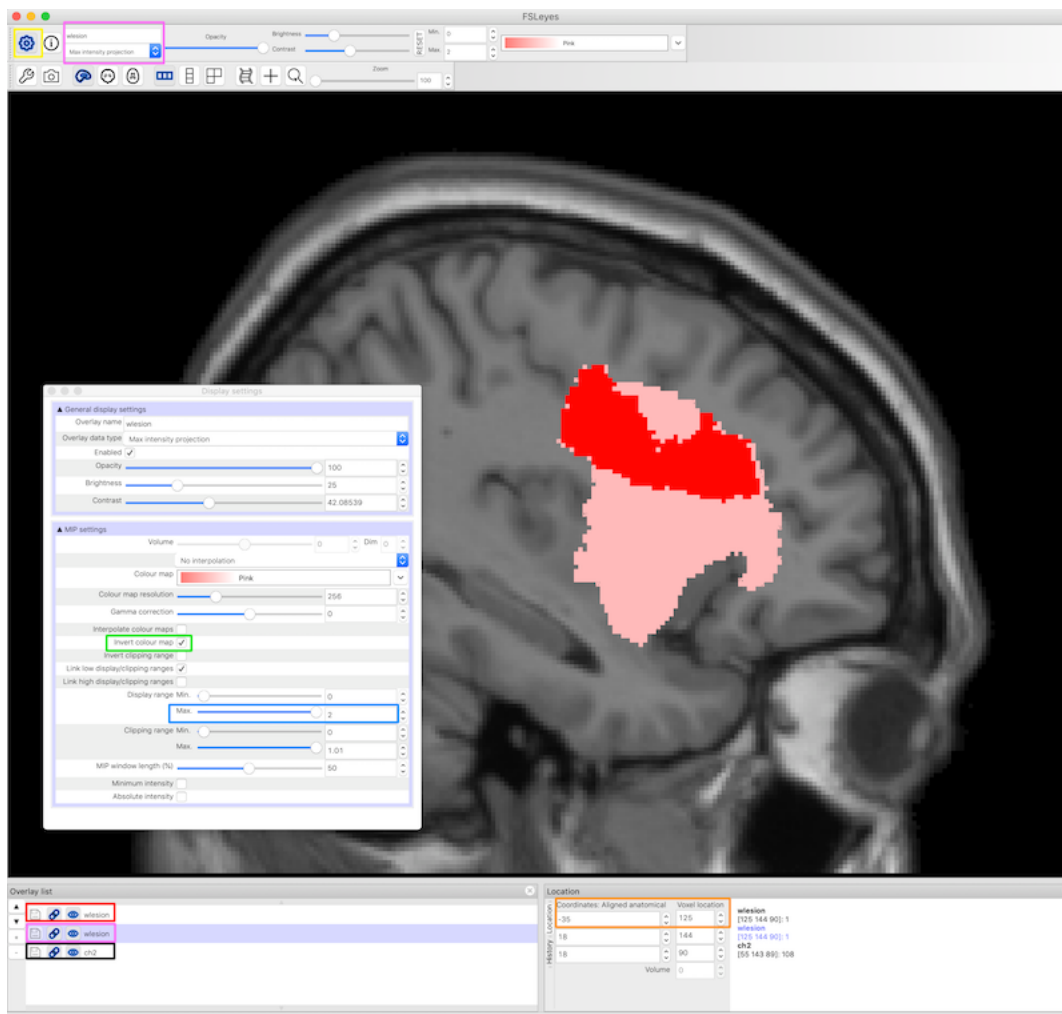
6.4.3 Convert Images from SPM MNI to FSL MNI

Not all MNI spaces are created equal. In 1 mm resolution, the SPM MNI image is 181x217x181 but the FSL MNI image is 182x218x182. This is not a lot, but is just enough to break some tools (like Tractotron) that expect one format and not the other. Here I provide a script to convert from the SPM12 MNI format to the FSL format. This script ensures that the shape, size and datatype of the image remain exactly the same, and, in fact, the coordinates of any structures remain exactly the same. I have Mark Jenkinson from the FSL group to thank for getting all the right flirt flags. You can download spm2fsl.sh

```
spm2fsl.sh lesion_mask_spm_1mm
```

Resting State

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2018_12_18
Date Updated: 2018_06_26
Tags: Neuroimaging software, func

7.1 Introduction

Resting state data is fMRI data collected while the subject is at rest, but not sleeping. This often involves staring at a fixation cross, but one group has devised a more engaging activity which nevertheless generates resting state networks. See the Inscapes article. This 7 minute movie can be shown to participants during a resting state scan. It is available on the task computer at the UofA scanner (on the desktop, and labeled 01_Inscapes_NoScannerSound_h264 1.mp4). Or you can download the original inscape movie with or without scanner sound.

7.1.1 FSL fMRI Resting State Seed-based Connectivity

Maintainer: Dianne Patterson, Ph.D. dkp @ email.arizona.edu
Author: Dr. Ying-hui Chou, Sc.D. for PSY 528 Cognitive Neuroscience, University of Arizona
Date Created: 2018_09_12
Date Updated: 2019_12_12
Tags: fMRI, resting state, SBC (seed-based connectivity)
Software: FSL 5.10 (or similar) with FSLeyes 0.26.1+ or greater
OS: UNIX (e.g., Mac or Linux)

Unix Primer

You will be using the unix command line to run FSL and interact with the directories and files you need for this tutorial. If you are unfamiliar with the Unix command line, you should complete a basic lesson, such as the one found here:

Unix Primer, especially *Section 1*

Goals

You will create a simple seed-based connectivity analysis using FSL. The analysis is divided into two parts:

1. Generate a seed-based connectivity map of the Posterior Cingulate gyrus for each of our three subjects.

2. Perform a group-level (a.k.a. higher-level) analysis across the three subjects.

Data

SBC.zip consists of preprocessed fMRI files in standard MNI space for three subjects who participated in a resting state fMRI experiment. Each fMRI file is a 4D file consisting of 70 volumes. You should download the data and unzip it:

```
unzip SBC.zip
```

The data should look like this:
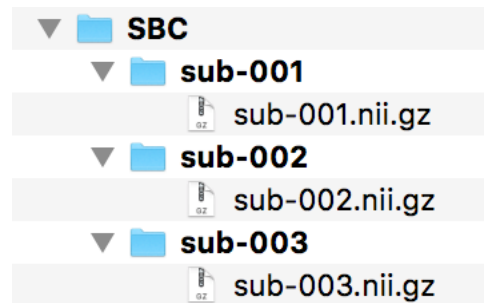
Under SBC, make a new directory called seed. You will need the seed directory to hold the Posterior Cingulate Gyrus mask you will create next.

Create Mask of Seed Region

The seed region will be the Posterior Cingulate Gyrus. You will identify the seed region using the Harvard-Oxford Cortical Atlas.

To start FSL, type the following in the terminal:

```
fsl &
```

N.B. The `&` symbol allows you to type commands in this same terminal, instead of having to open a second terminal.
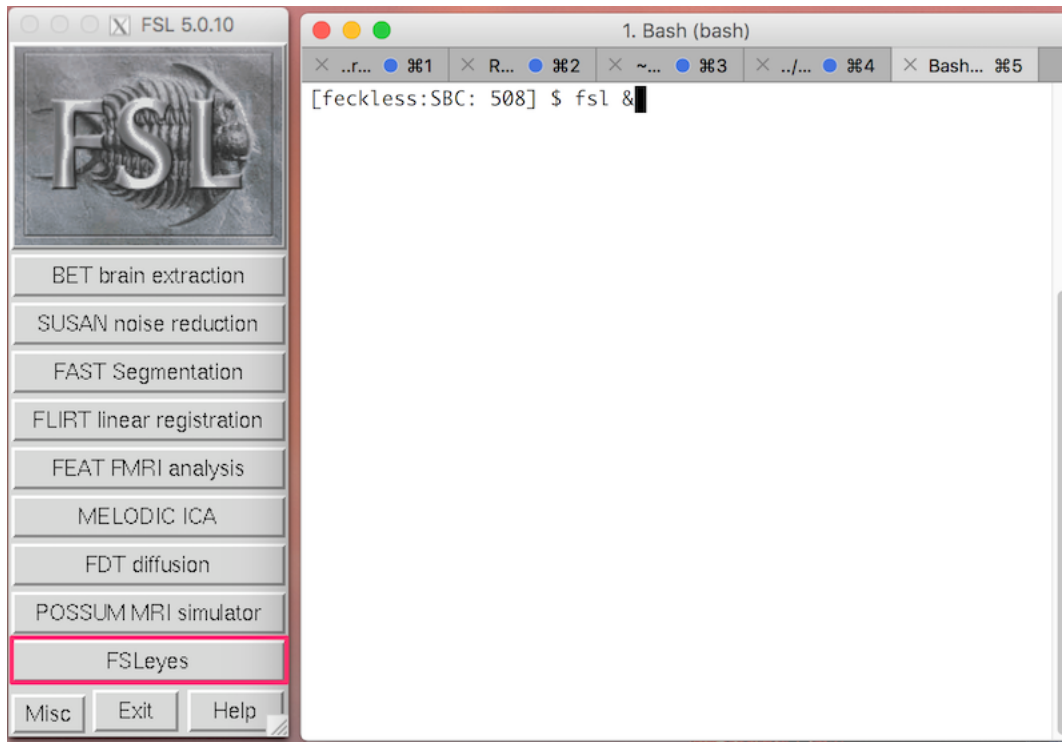
- Click FSLeyes to open the viewer.
- Click *File Add standard* Select *MNI152_T1_2mm_brain.nii.gz* Click *Open*.
- You will see the standard brain image in three different orientations.
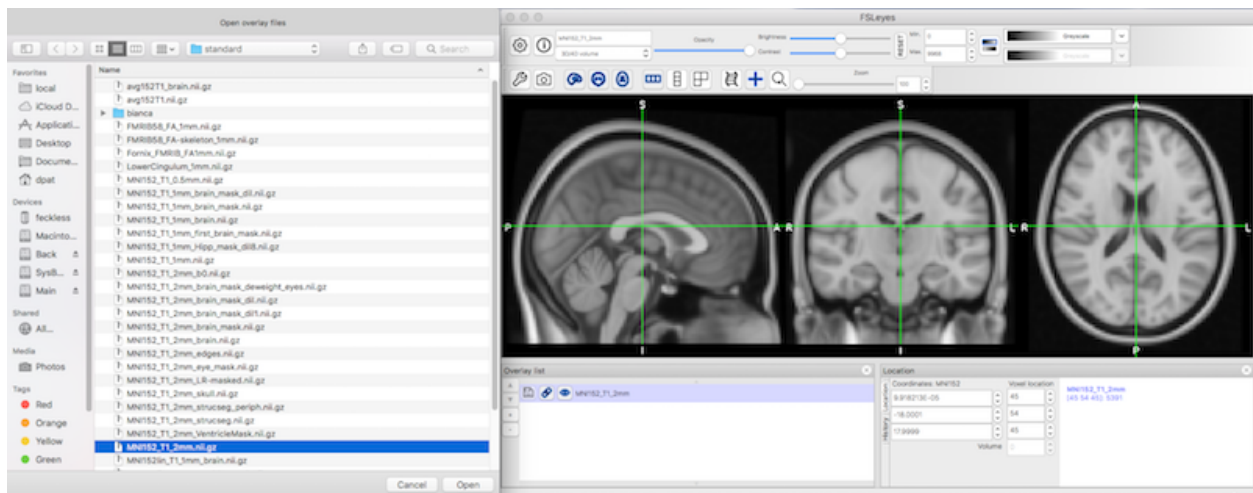
- Click *Settings Ortho View 1 Atlas panel*
- An atlas panel opens on the bottom right.
- Select *Atlas search* (red box, lower middle).
- Insure *Harvard-Oxford Cortical Structural Atlas* is selected (purple highlighting).
- Type `cing` in the search box and check the *Cingulate Gyrus, posterior division* (lower right) so that it is overlaid on the standard brain.
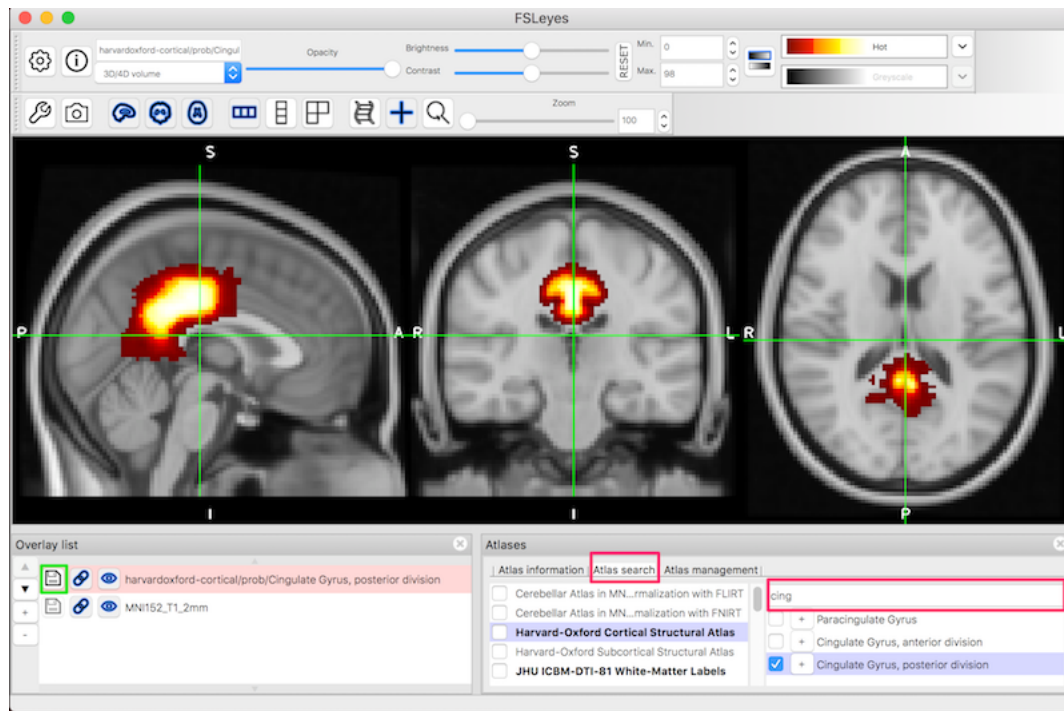
- To save the seed image, click the save symbol (green box, bottom left) next to the seed image.

- Save the image as *PCC* in the seed directory. If you go to the seed directory, you will see PCC.nii.gz in the folder.

- Leave FSLeyes open.

In the terminal, cd to the seed directory. You are going to binarize the seed image so we can use it as a mask:

```
fslmaths PCC -thr 0.1 -bin PCC_bin
```

- In FSLeyes click *File   Add from file* to compare PCC.nii.gz (before binarization) and PCC_bin.nii.gz (after binarization).

- You can close FSLeyes now.

Extract Time Series from Seed Region

For each subject, you want to extract the average time series from the region defined by the PCC mask. To calculate this value for sub-001, type:

```
fslmeants -i sub-001 -o sub-001_PCC.txt -m ../seed/PCC_bin
```

Repeat for sub-002 and sub-003.

You can view the timecourse in the text file using FSLeyes:

- Display sub-001.nii.gz in FSLeyes (*File → Add from file → sub-001.nii.gz*)

- On the FSLeyes menu, click *View → Time series*. You see the time series of the voxel at the crosshairs.

- Move the crosshairs off the brain, so the time series is a zero-line.

- Click *Settings → Time series 2 → Import* and select the *sub-001_PCC.txt* file you just created.

- Click *OK*. You should see the timecourse of the txt file. You can add text files for sub-002 and sub-003 if you wish.

- You can close FSLeyes when you are done.
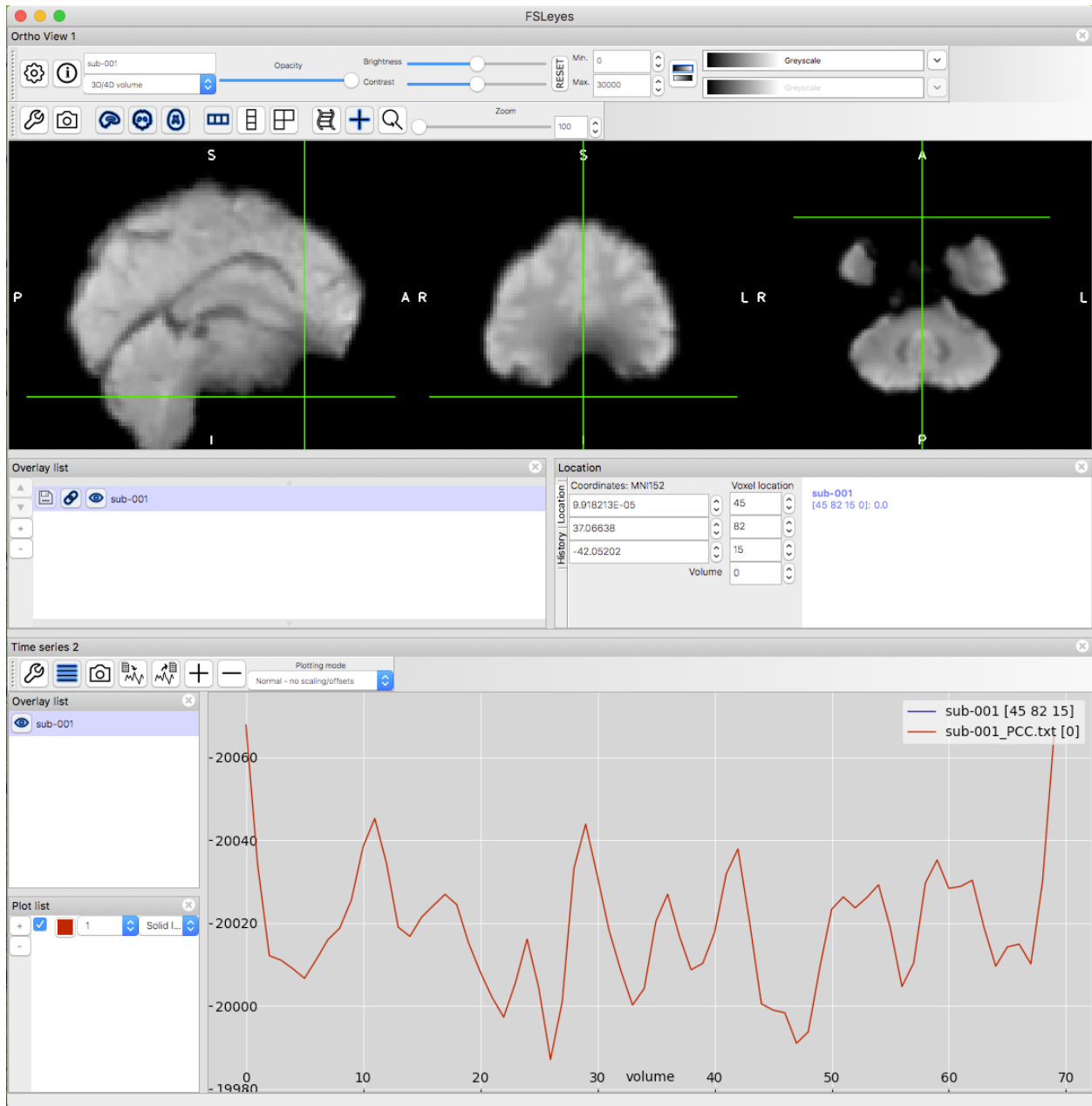
Run the FSL FEAT First-level Analysis

For each subject, you want to run a first level FEAT analysis showing us the brain regions that have activity correlated to the mean PCC activity.

Start FSL again:

```
fsl &
```

Select *FEAT FMRI analysis*.
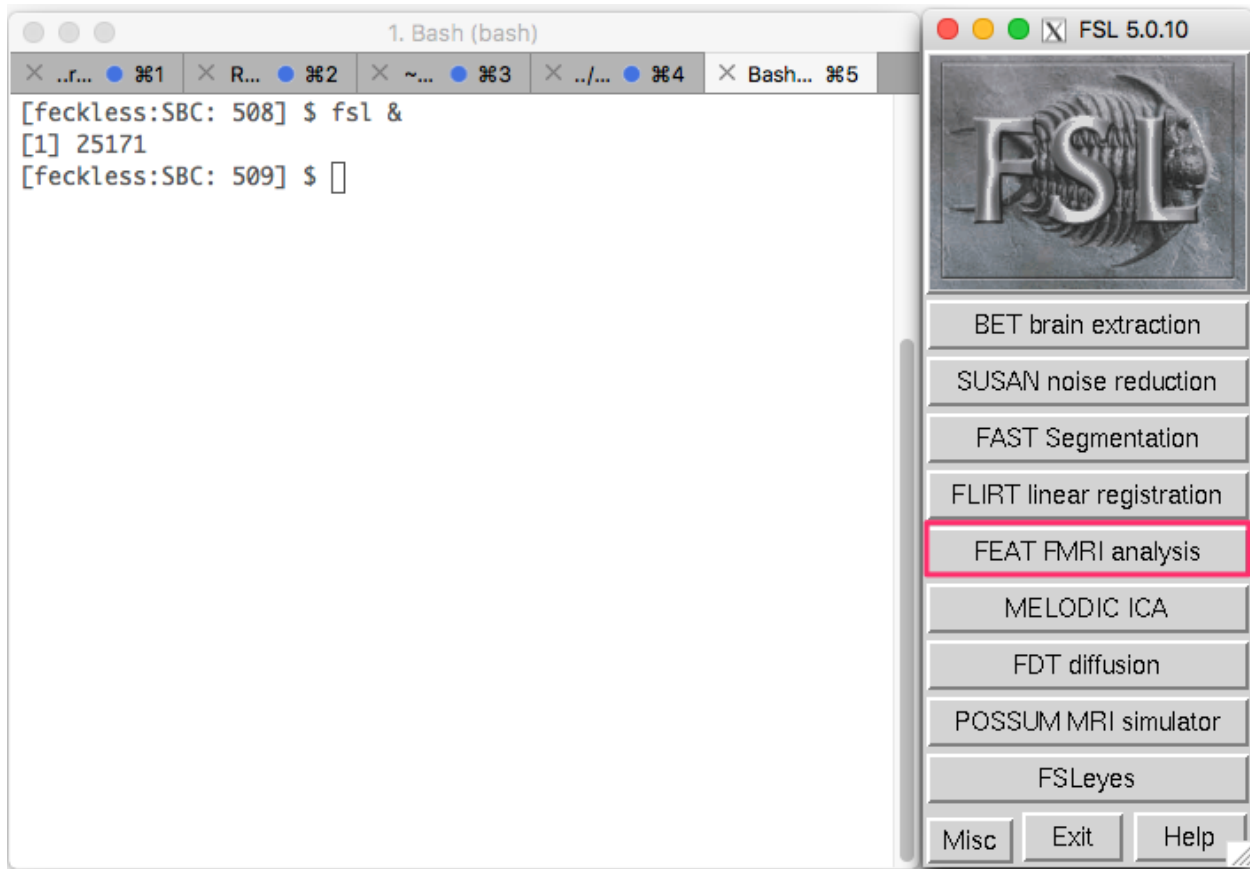
- This is *FEAT*, the *FMRI Expert Analysis Tool*.
- On the default *Data* tab, *Number of inputs* is 1.
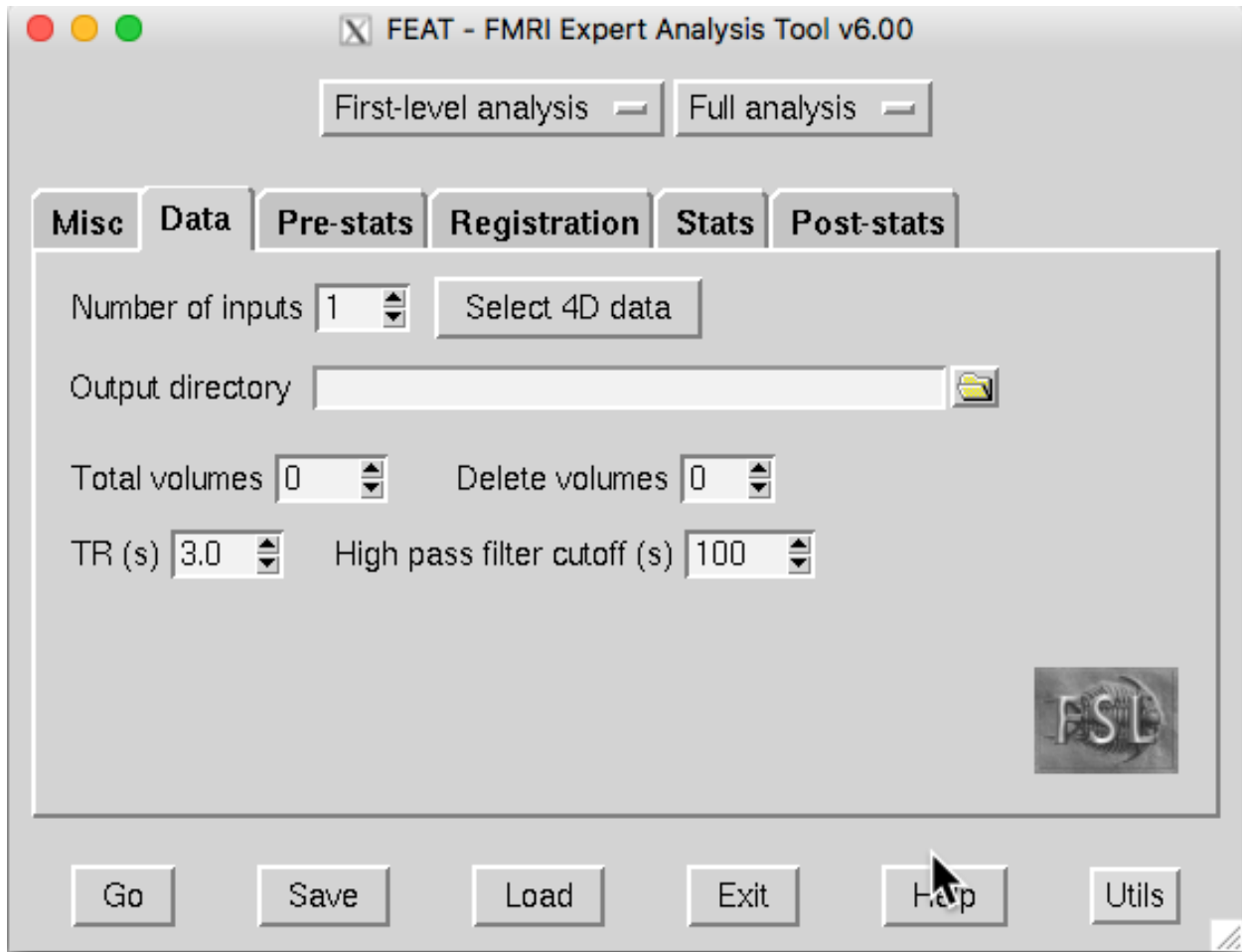
- Click *Select 4D data*, and a *Select input data* window will appear.
- Click the yellow folder on the right-hand side and select sub-001.nii.gz.
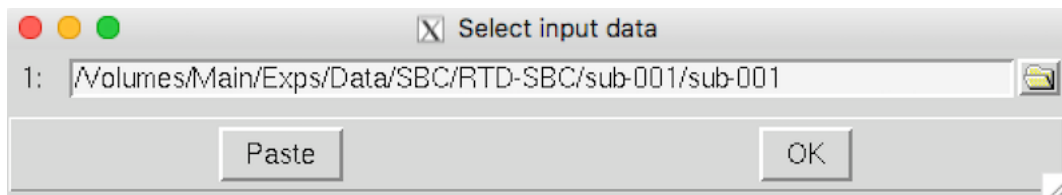
Click *OK* to close each Select input data window.
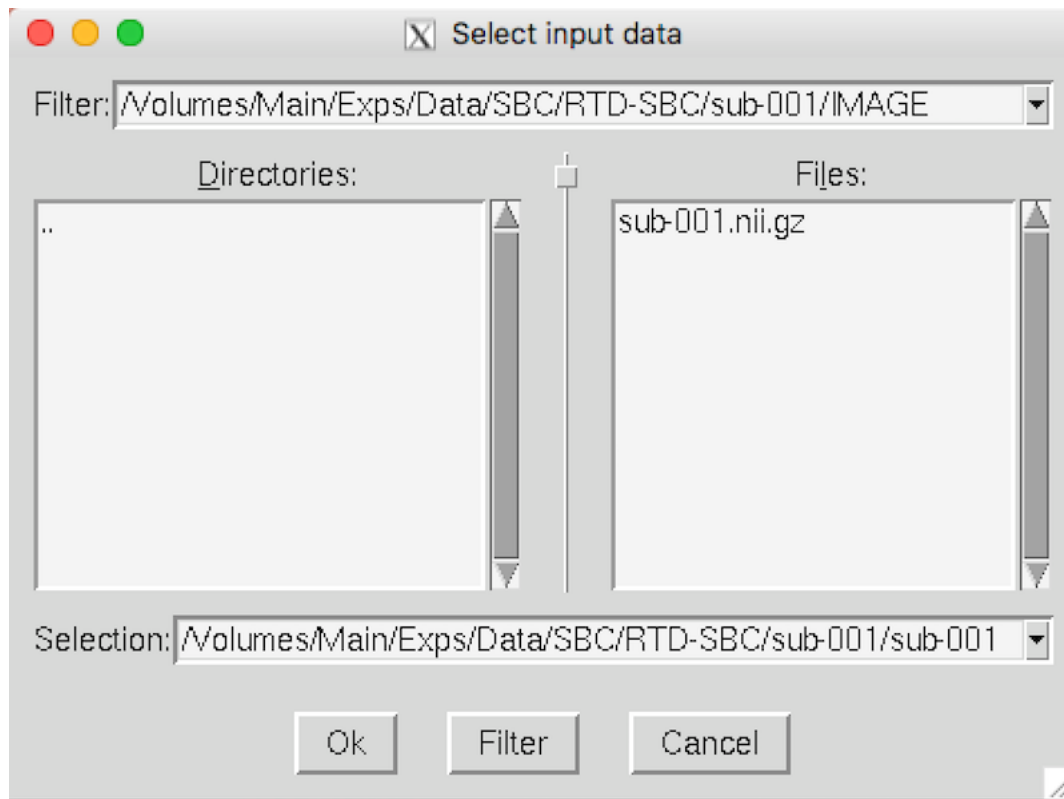
If you see the following message, click *OK*.
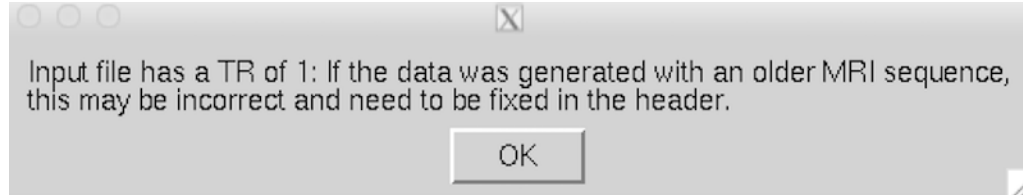
You are back to *FEAT* – FMRI Expert Analysis Tool* window.

- Click the yellow folder to the right of the *Output directory* text box.

- A window called *Name the output directory* will appear.

- Go to the sub-001 folder and click *OK*.

- You have told FSL to create a feat directory called sub-001.feat at the same level as the subject and seed direc-
  tories.

- The *Data* tab should look like this:
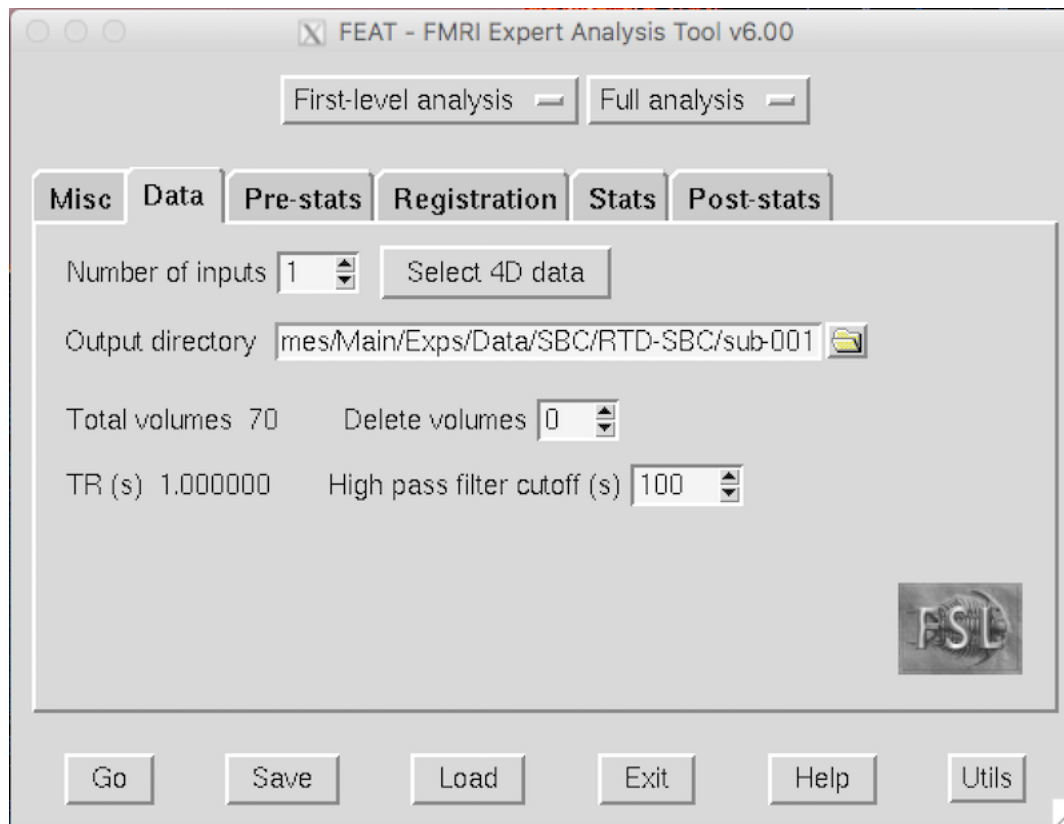
Click the *Pre-stats* tab. Because this dataset was already preprocessed, you don't need most of these options. Change the parameters to match the picture:
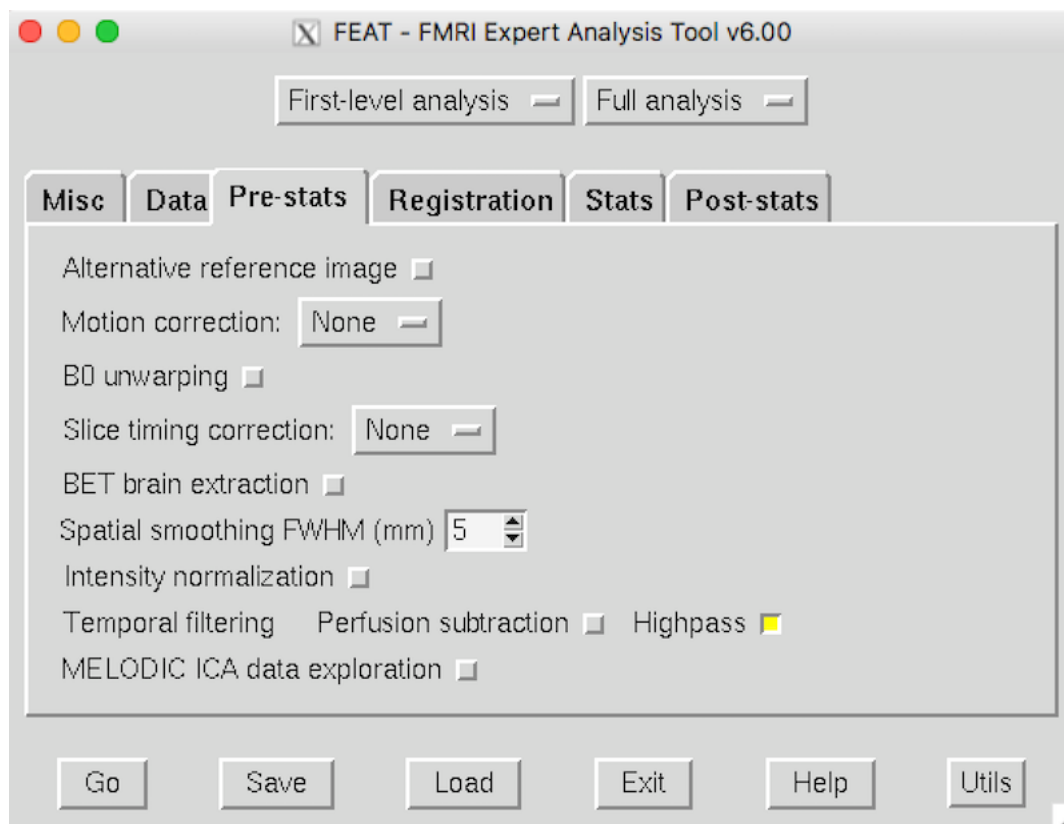
Click the *Registration* tab and make sure it matches this picture:
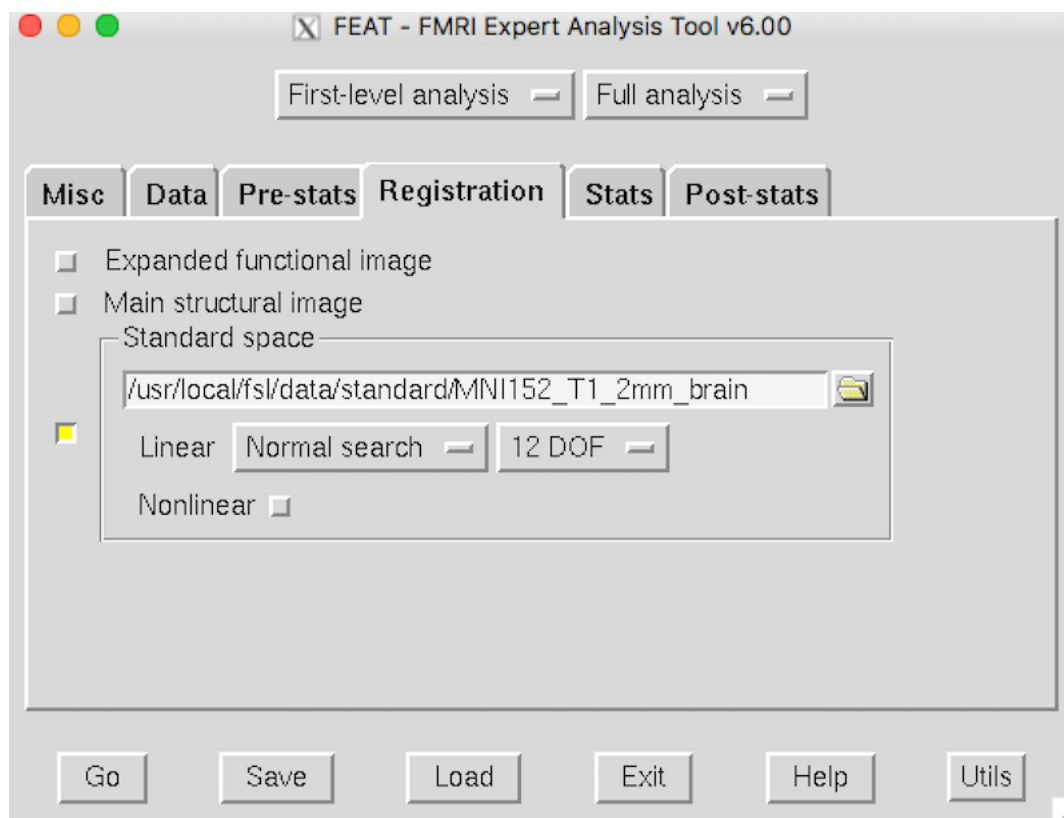
Click the *Stats* tab and then *Full model setup*:
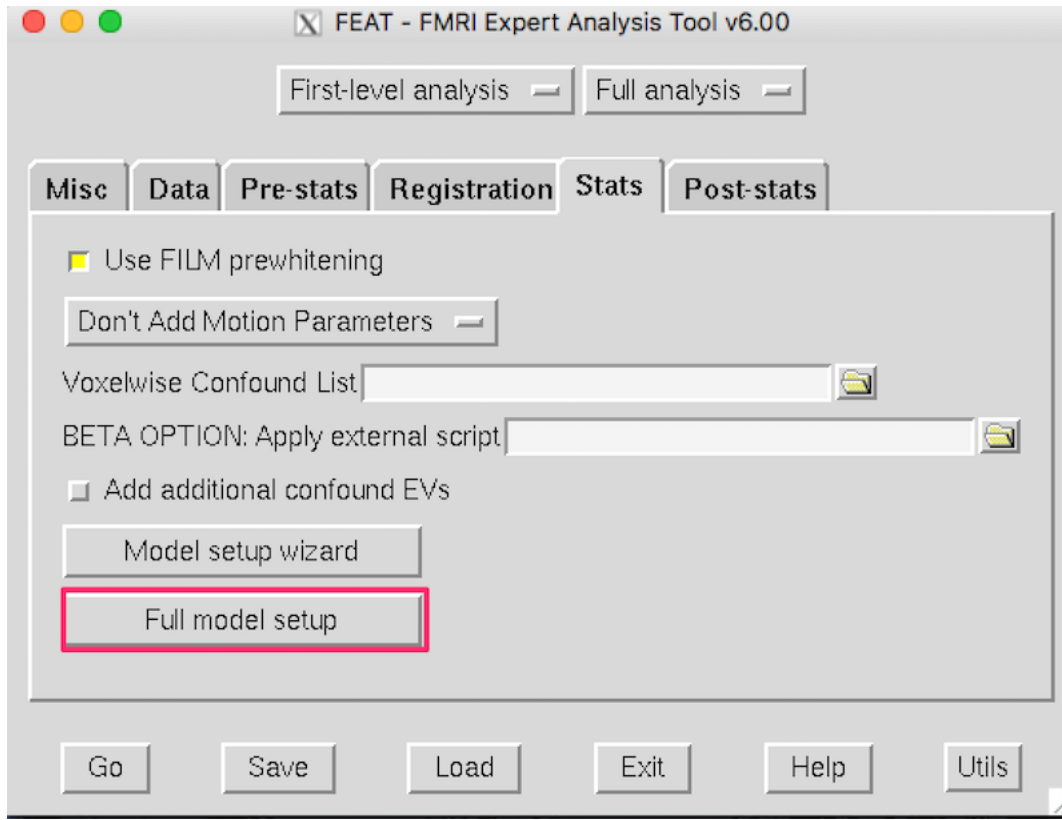
A *General Linear Model* window will appear.

- The *Number of original EVs* is 1.

- Type `PCC` for the EV name.

- Select *Custom (1 entry per volume)* for the *Basic shape*.

- To the right of the *Filename* text box, click the yellow folder and select *sub-001_PCC.txt* This is the mean time series of the PCC for sub-001 and is the statistical regressor in our GLM model.

- The first-level analysis will identify brain voxels that show a significant correlation with the seed (PCC) time series data.

- Select *None* for *Convolution*, and deactivate *Add temporal derivate* and *Apply temporal filtering*

- Your general linear model should look like this:
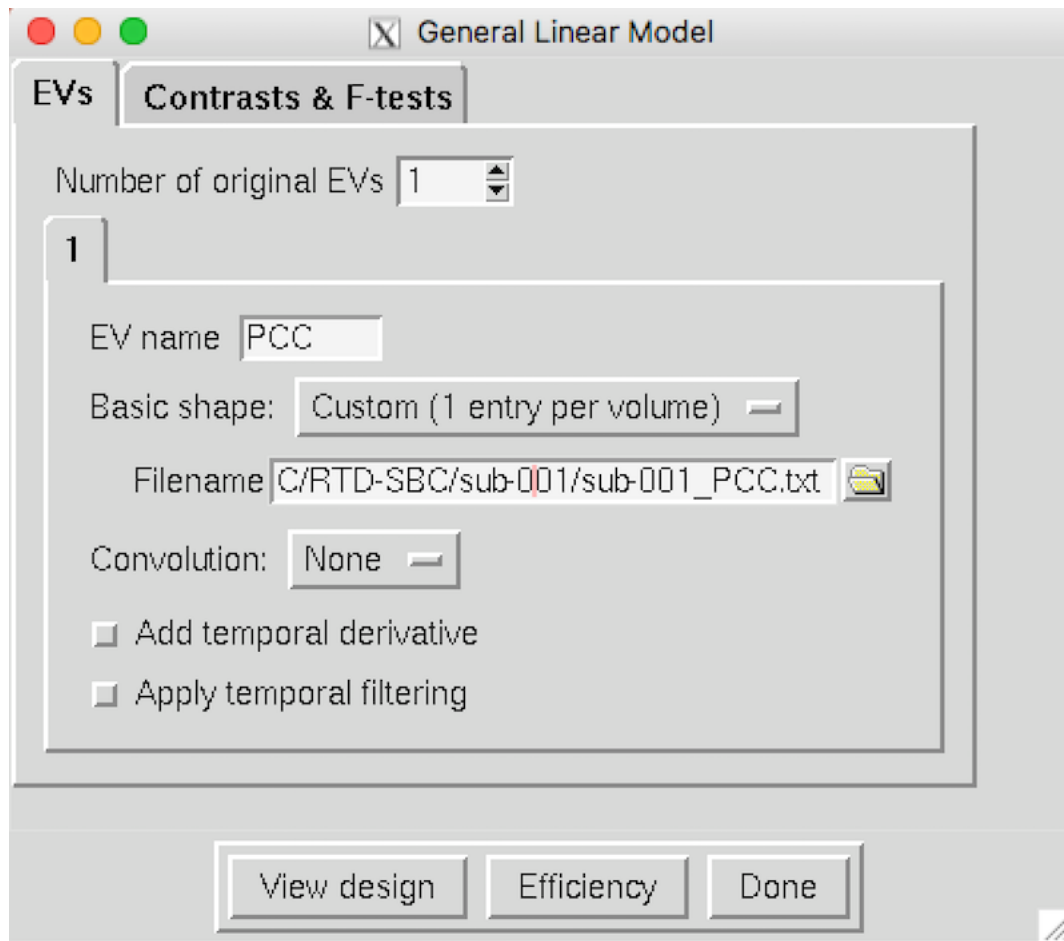
- In the same General Linear Model window, click the *Contrast & F-tests* tab.

- Type `PCC` in the *Title*, and click *Done*.

- A blue and red design matrix is displayed. You can close it.

- You don't need to change anything in the *Post-stats* tab.

- You are ready to run the first-level analysis. Click *Go* now.

- A FEAT Report window will appear in your web browser. You will see *Still running*. Wait 5-10 minutes for the analysis to finish.

- After the analysis finishes, you see a FEAT Report in the browser. Click *Post-stats*, and you should be able to see the following results:
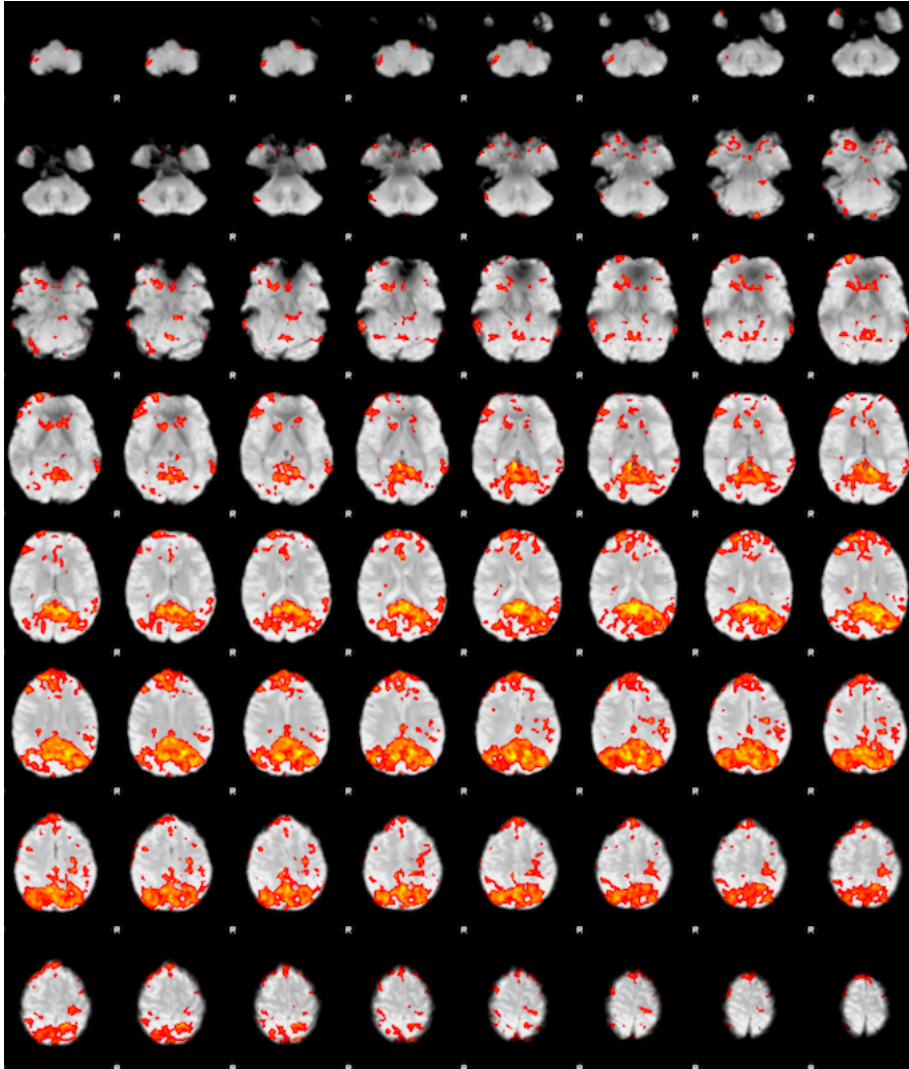
- Click the figure to see the cluster list and the local maxima data.

- You have finished the first-level analysis for sub-001. You can save your model by clicking *Save*.

- To run a group analysis, you need at least three datasets. You must repeat the above steps for sub-002 and sub-003. There are three values you need to modify:

1. **The input file on the *Data* tab:** For example, click Select 4D data, and change the input data from sub-001/sub-001.nii.gz to sub-002/sub-002.nii.gz

2. **The output directory on the *Data* tab:** For example, change the output directory from sub-001 to sub-002.

3. **The PCC txt file on the *Stats* tab → Full model setup:** For example, change the filename from sub-001/sub-001_PCC.txt sub-002/sub-002_PCC.txt.

- Once you finish the first-level analysis for sub-002, you can expect to see the following results:
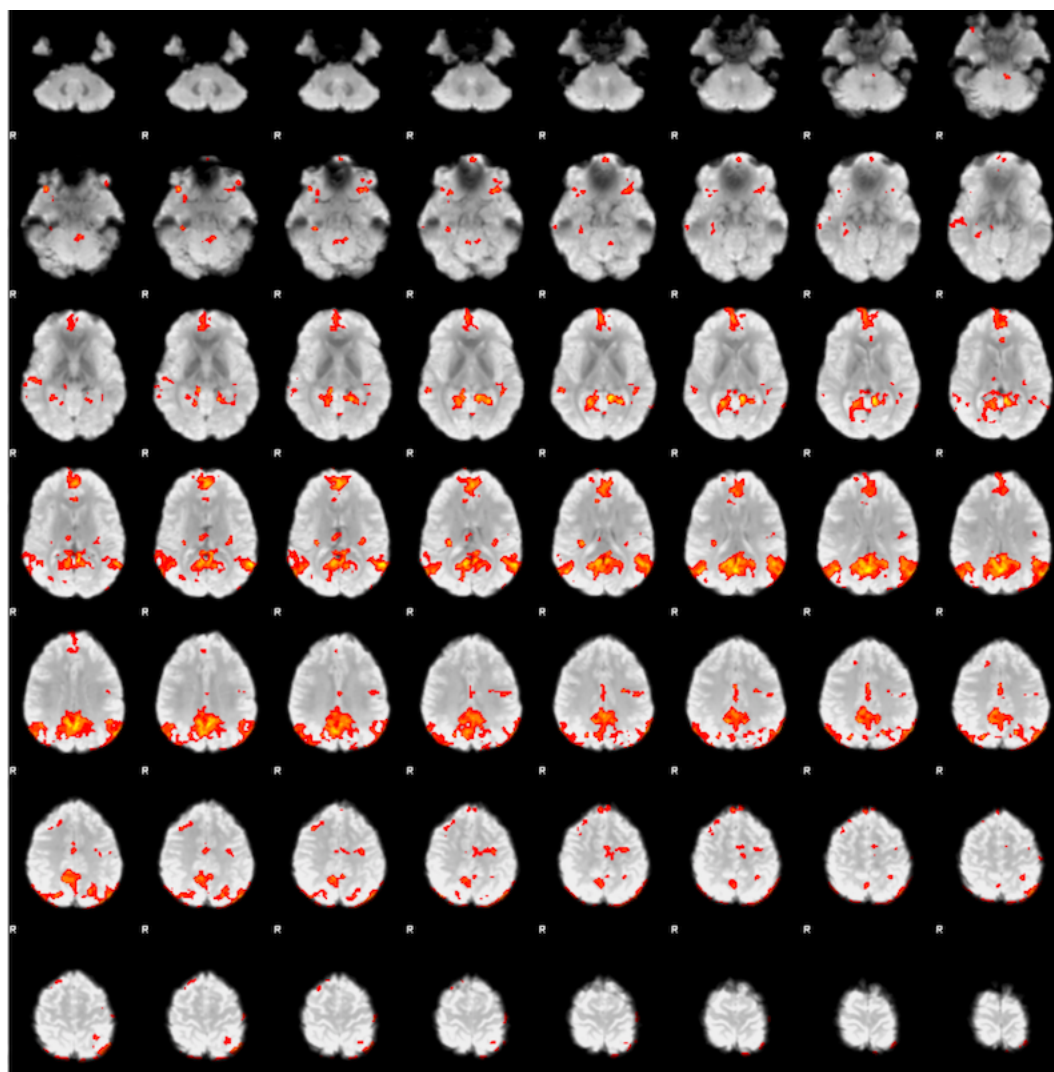
Repeat the same steps for sub-003.
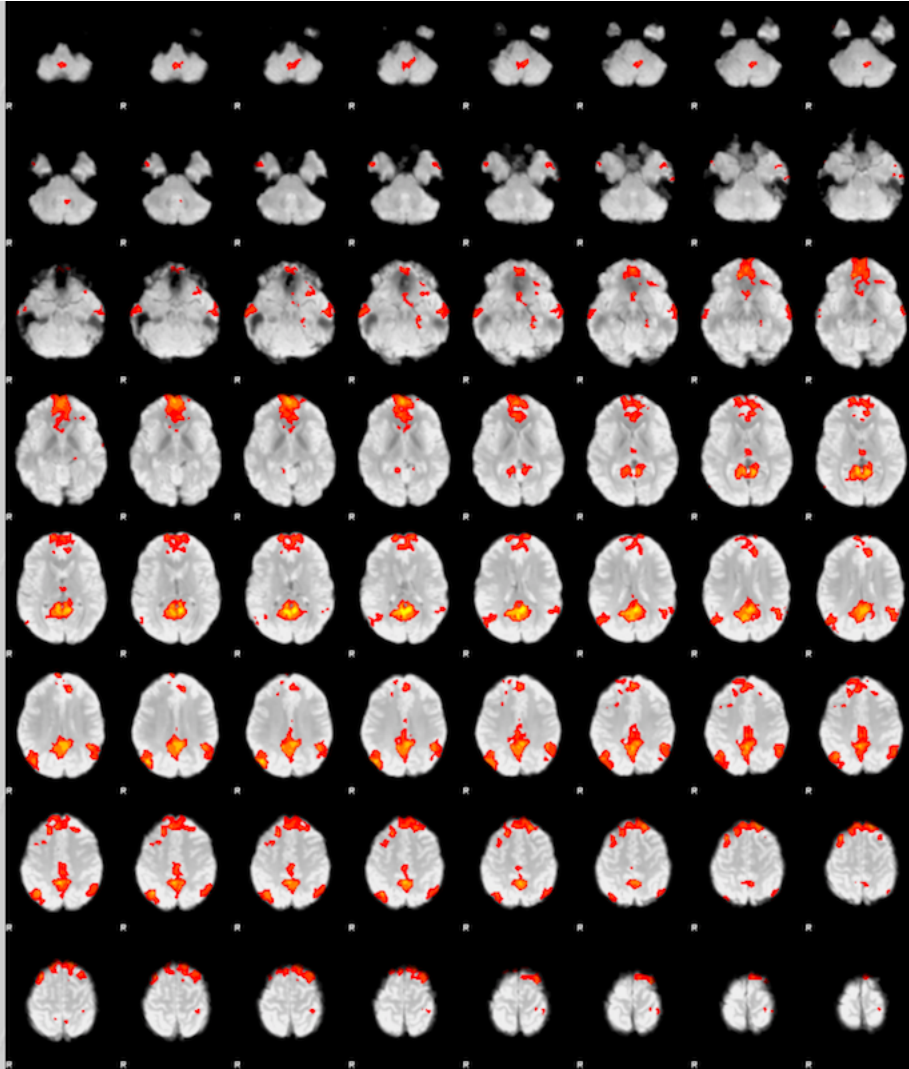
The FSL FEAT Higher-level Analysis

In the *FEAT – FMRI Expert Analysis Tool Window*, select *High-level analysis* (instead of *First-level analysis*). The higher level analysis relies on:

1. Each of the individual subject feat analyses AND

2. A description of the GLM model.

Select Each Individual Subject FEAT Analysis

On the *Data* tab:
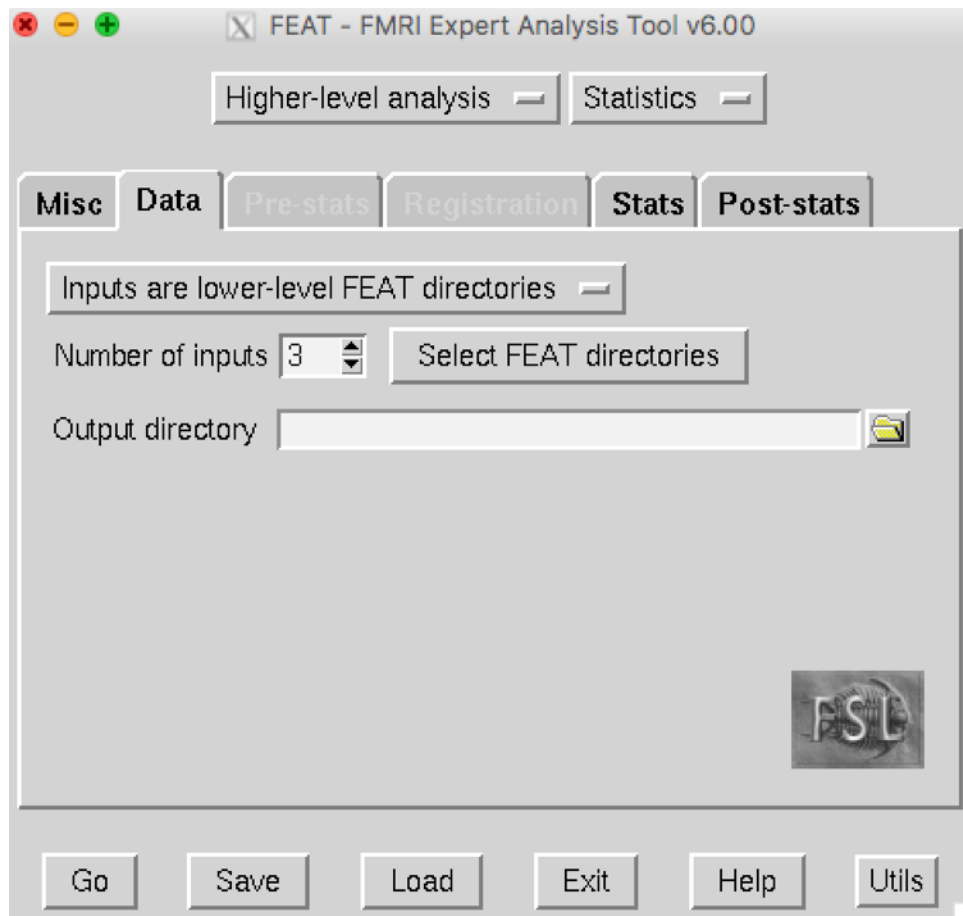
- Select the default option, *Inputs are lower-level FEAT directories*.

- *Number of inputs* is 3.

- Click the *Select FEAT directories*. A *Select input data* window will appear:
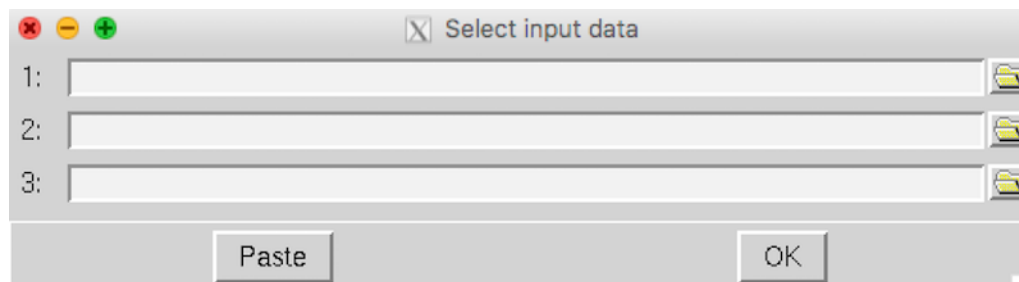
- Click the yellow folder on the right to select the FEAT folder that you had generated from each first-level analysis.

- After selecting the three Feat directories, it will look like this:
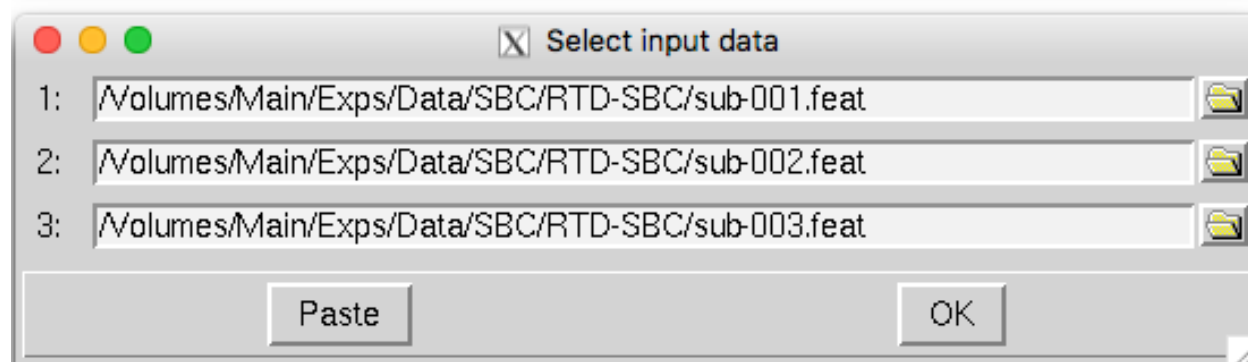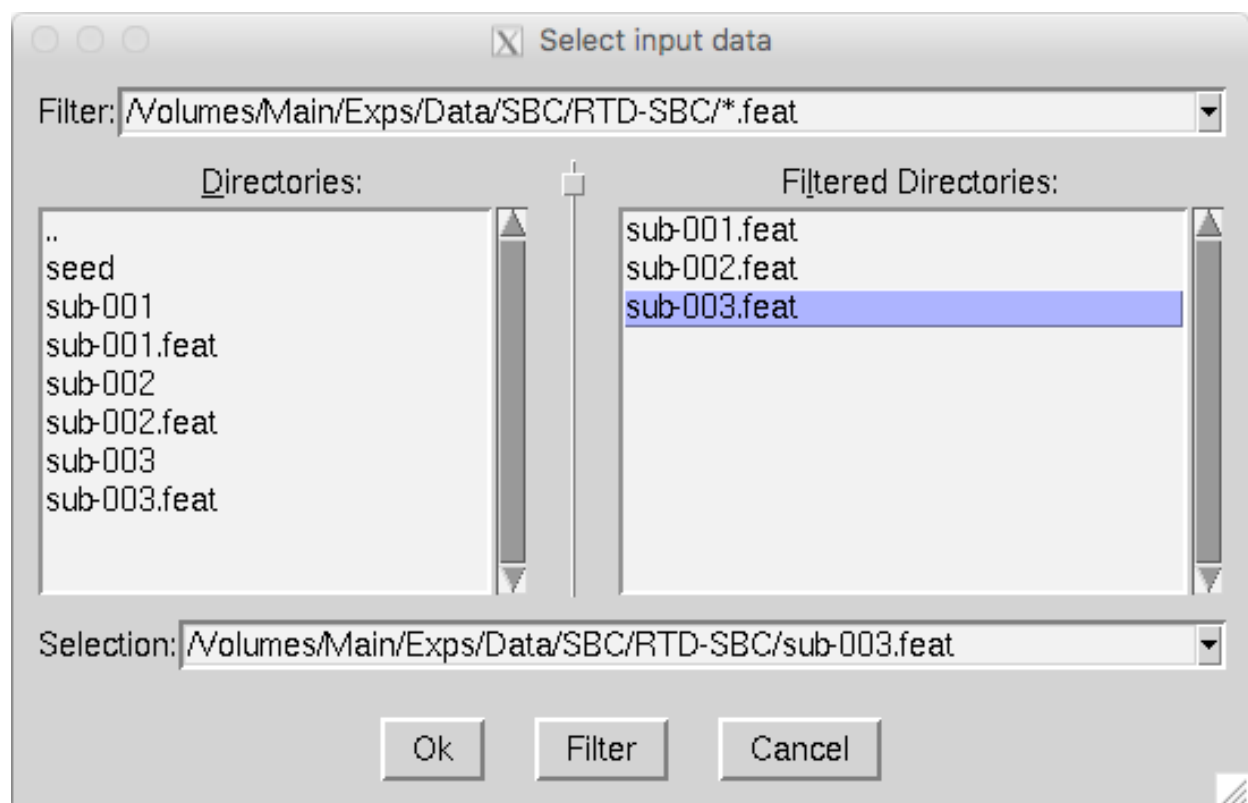
Click *OK*.

Name an Output Directory

- Click the yellow folder to the right of the Output directory text box, and provide a name for the output directory, e.g. `PCC_group` (N.B., Don't put *PCC_group* inside a subject directory, or you'll have a hard time finding it later).

- Click *OK*.
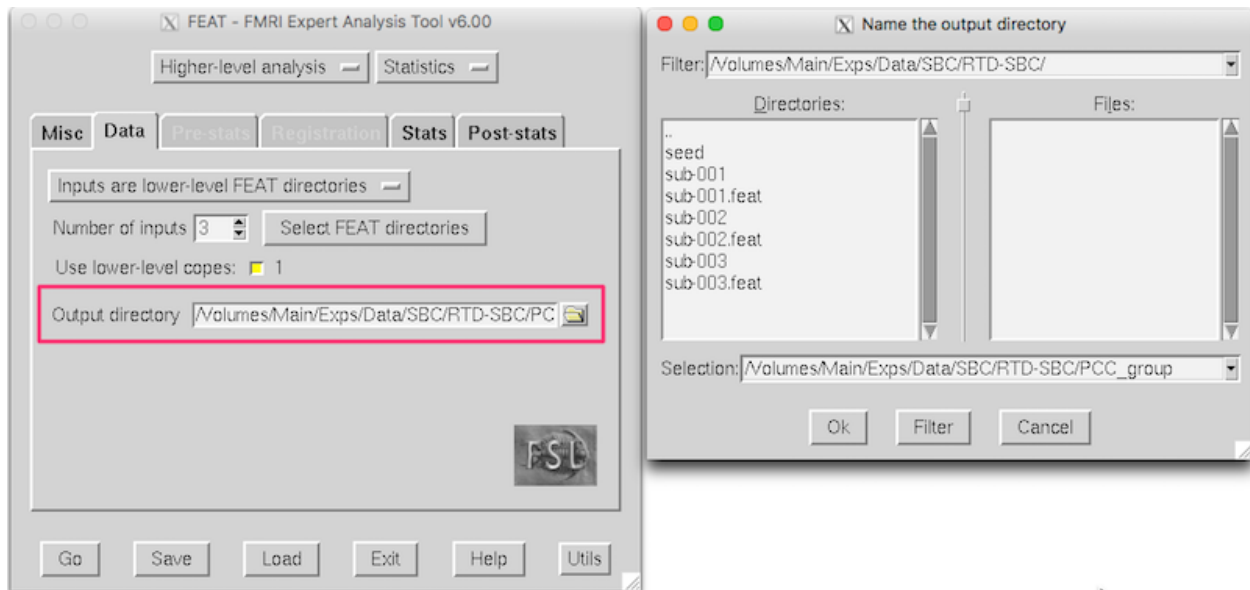
On the *Stats* tab, click *Full model setup*. In the *General Linear Model* window, name the model `PCC` and insure the interface looks like this:
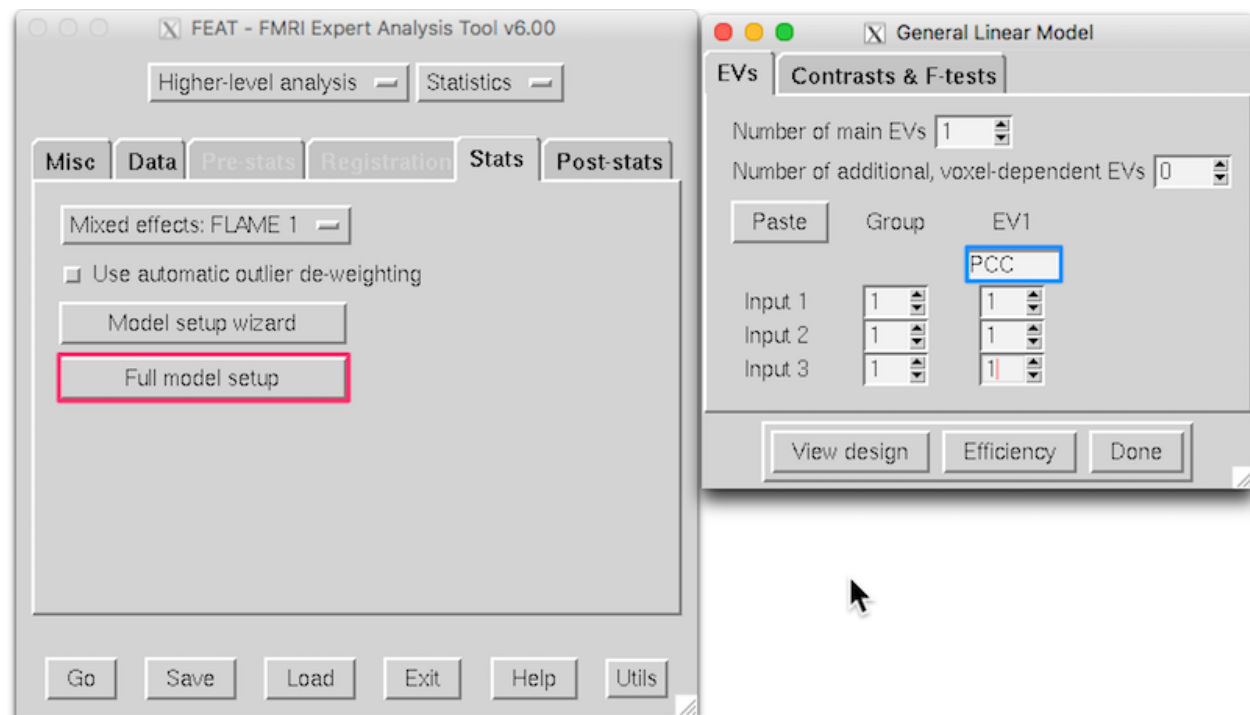
- Click *Done* to close the *General Linear Model* window.

- Because there are so few subjects here, and you want to see some results, you are going to set z down to 2.3 instead of 3.1 on the FEAT *Post-stats* tab:
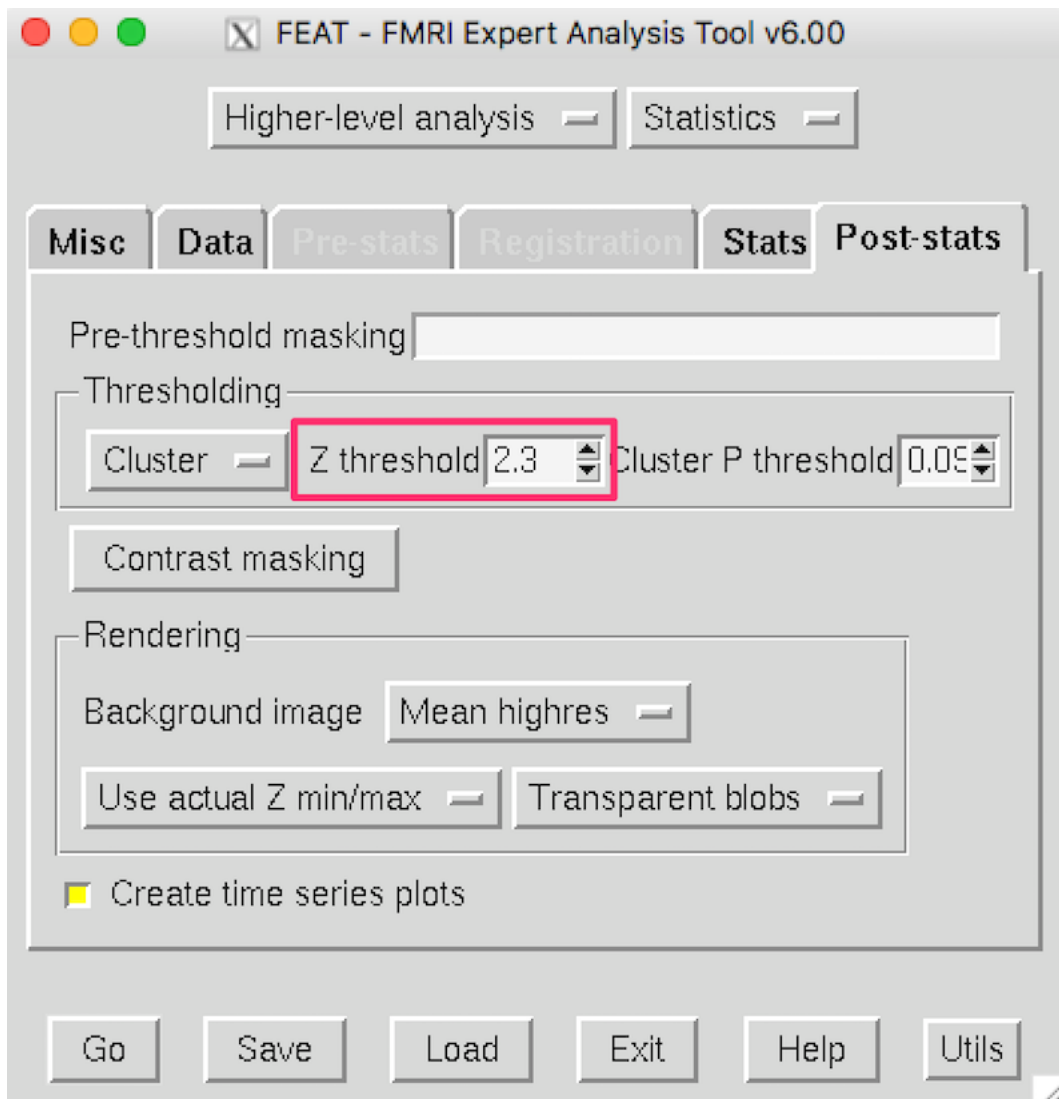
Run the Higher-level Analysis

- Click *Go* to run the Higher-level analysis.

- While you see *Still running*, you will need to wait a few minutes for the analysis to be done.

- After 5-10 mins, click *Results*.

- Then click *Results for lower-level contrast 1 (PCC)* if it is available, that means it is done. * Click *Post-stats*, and you should see this:
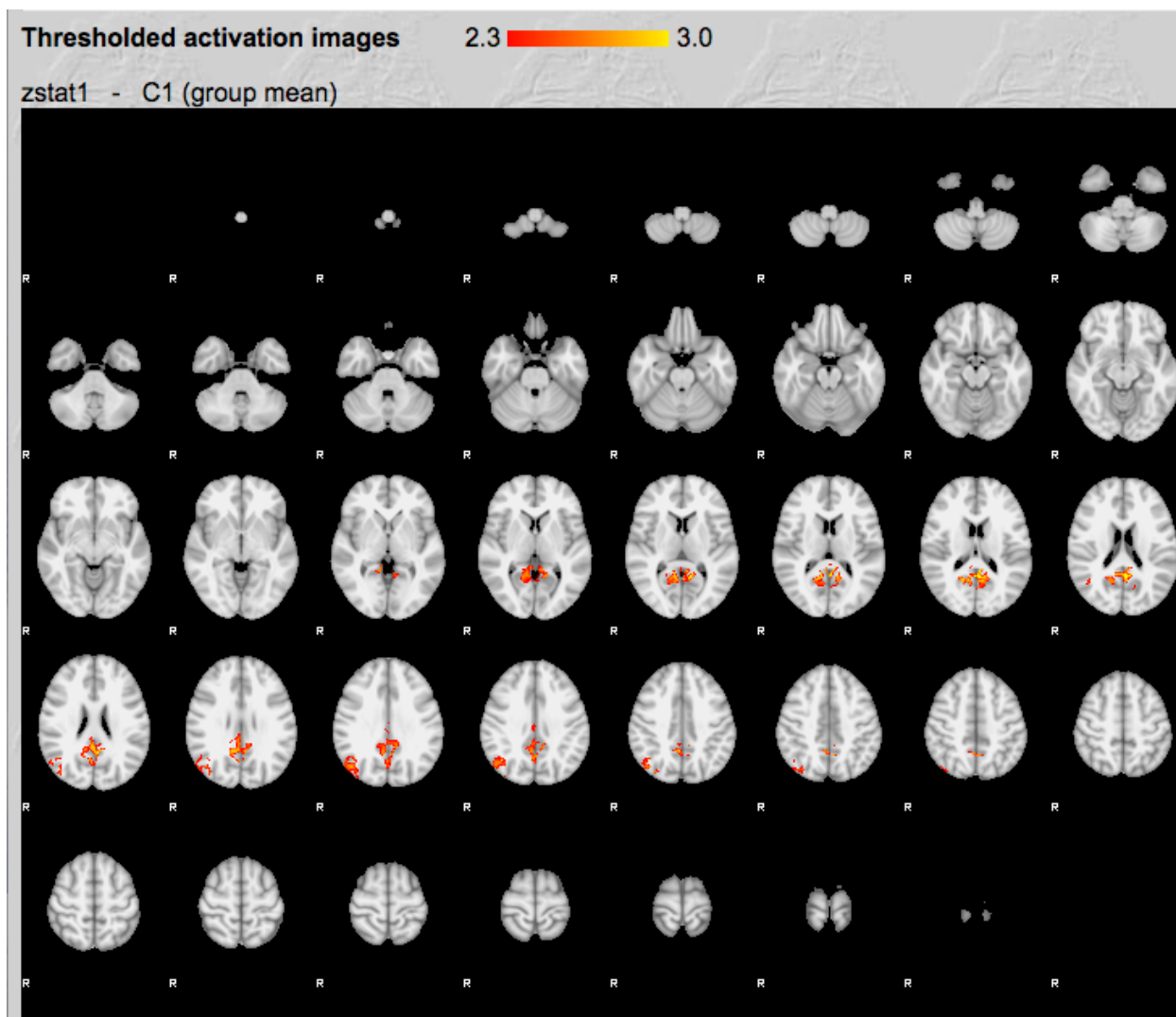
As you can imagine, due to the small sample size (N=3), many voxels do not pass the multiple comparisons correction. You would see a more complete default mode network with more subjects.

Stroke Lesions

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2018_06_14
Date Updated: 2019_07_26
Tags: anat, lesion, stroke, segmentation, CT, header editing, registration, normalization

The focus of these pages is large lesions, such a stroke lesions. These large lesions present problems not encountered with smaller punctate lesions (e.g., MS lesions):

1. *Creating the Lesion Mask*: Large stroke-related lesions are difficult to characterize. Manual methods are laborious and the inter-rater reliability hovers around 80%. Automated routines perform poorly. We explore semi-automated pipelines that can help with both the speed and consistency of drawing lesion masks.

2. *Working with Clinical Data*: Although we use a large homogenous dataset to test normalization techniques, real stroke studies may have more heterogenous data. For example, stroke patients who cannot have MRI scans may sometimes have CT scans. Working with the available imaging data presents unique challenges.

3. *Lesion Normalization*: A large area of abnormal tissue can adversely affect how the image is warped into standard space. This can result in lesion shrinkage, as discussed in Andersen, S. M., Rapcsak, S. Z., & Beeson, P. (2010). Cost function masking during normalization of brains with focal lesions: Still a necessity? NeuroImage.. Here I explore normalization in a homogenous dataset comparing the results of using different tools and parameters.

4. **Statistical Analysis of Lesions**: Both volume and location of the lesion should be considered in statistical analyses (otherwise there is the danger of simply discovering that big lesions damage more behaviors).

8.1 Creating a Lesion Mask

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2019_06_25
Date Updated: 2019_08_17
Tags: anat, lesion, stroke, drawing masks, segmentation, header editing, registration

It is difficult to draw lesion masks consistently. Liew et al. (2018) report that "inter-rater reliability was 0.76 ± 0.14, while the intra-rater reliability was 0.84 ± 0.09". Clearly rater reliability is THE major source of variance in lesion masks. We know that some lesions are more difficult to delineate, and we can hypothesize that large lesions are more heterogeneous and more likely to intersect brain and ventricle boundaries, which might make them more difficult to characterize consistently. We also know that acute lesions are more difficult to characterize than chronic lesions (Pelagie Beeson, Ph.D, Personal communication).

A traditional approach to creating lesion masks is provided by the ATLAS project and can be found here: MRIcron Lesion Tracing

> **Warning:** MRIcron assumes isotropic (or near isotropic) voxels. This means it is not well suited to displaying clinical data which often has a high resolution in-plane, but very thick slices. Although ITK-SNAP can display anisotropic data, the semi-automated routines benefit from isotropic data. See *Correct Extreme Anisotropy* below for a description of how to correct anisotropy.

A semi-automated approach using ITK-SNAP can be found here *ITK-SNAP Segmentation: Lesions*. The benefits of ITK-SNAP over MRIcron are described in the Introduction here: *ITK-SNAP*.

8.1.1 iPad Drawing Tools

It is difficult to draw for very long with a mouse or trackpad. Wacom tablets are expensive. I have found the following two iPad options to be useful. Both have pros and cons. Both are better with the Apple pencil.

- The *iMango* software is a standalone application for the iPad. $15.
- Another alternative is Astropad Standard $30.
    - Astropad requires a connection (wired or wireless) to your primary Mac (which assumes you HAVE a mac).
    - However, Astropad displays any Mac application for use on the iPad.
    - If you like MRIcron or ITK-SNAP, then Astropad provides a inexpensive competitor to the Wacom tablet.

8.1.2 Related Pages

ITK-SNAP Segmentation of Stroke-related Lesions

Maintainer: Dianne Patterson, Ph.D. dkp @ email.arizona.edu
Author: Dr. Aneta Kielar, Ph.D. Speech, Language and Hearing Sciences, University of Arizona
Date Created: 2018_10_24
Date Updated: 2019_08_19
Tags: lesion, segmentation, anat
Software: ITK-SNAP, FSL (you can do the first part of the tutorial with ITKSNAP, even if you don't have FSL)
OS: Windows, Mac or Linux

Introduction

What is a Stroke-related Lesion?

A chronic stroke-related lesion is a heterogeneous CSF-filled cavity surrounded by a rim of damaged tissue. Lesions vary in shape and size. It is often challenging to work with lesion data because other brain structures, especially ventricles, can change size and shape after the brain injury. On a T1-weighted image, CSF is very dark and the associated area of neuronal damage is darker than surrounding healthy tissue. However, the damaged tissue may be very similar to grey matter, making identification of the lesion even more difficult. Both the CSF-filled cavity and the associated damaged tissue should be considered lesion. In addition, we include ventricular enlargement as part of the lesioned tissue if the lesion and ventricle are contiguous as in the tutorial data.

What is a Lesion Mask?

It can be valuable to mask lesions. Lesion masking involves creating a layer that covers the lesion (and only the lesion). In the picture below, you see an axial slice through a brain with a large lesion. On the right, we have covered the lesion with a red mask (translucent so you can see what is underneath). Most often we mask lesions manually. Lesion masks are sometimes referred to as *lesion maps* or as *segmentations*.

Manual masking of lesions can take hours or days because we need to outline the lesion carefully on each slice of the image. But this mask took only a few minutes to generate in ITKSNAP, and although it will need to be edited, it is a good start.
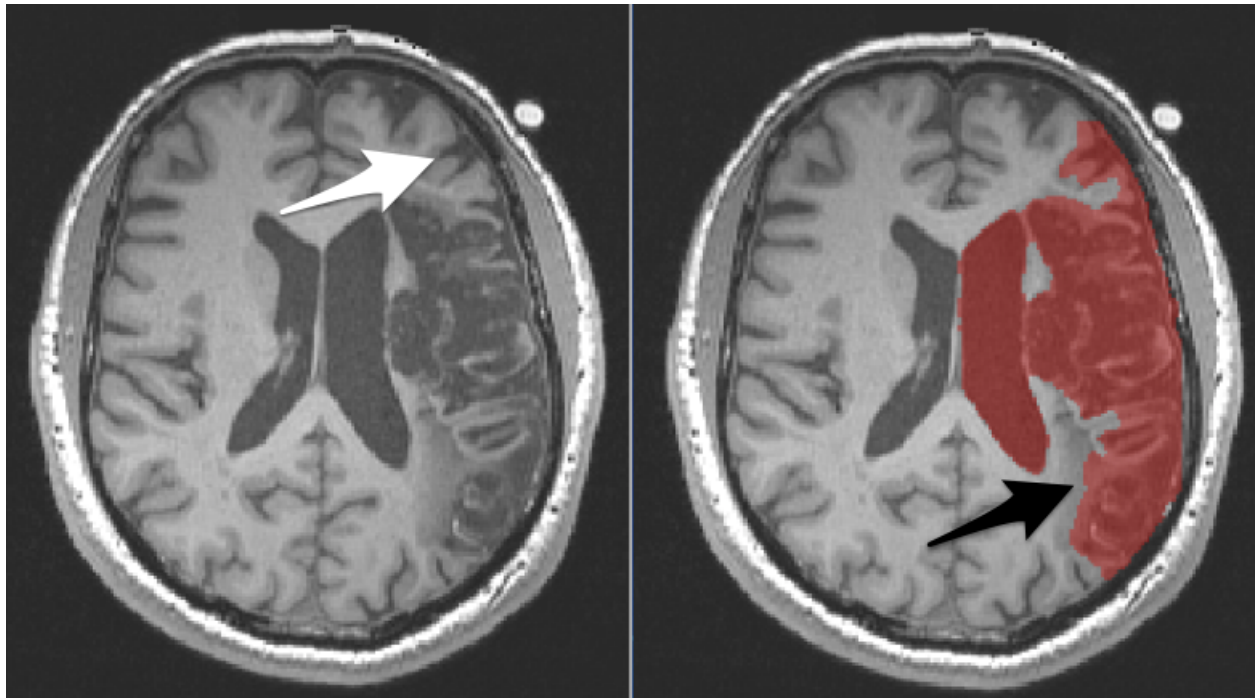
To determine whether an area should be included in the lesion mask, compare the tissue on the damaged side to the normal tissue in the other hemisphere.

- There are two arrows on the above images:
  - **Left** The white arrow points to a region that is incorrectly covered by the mask. This is a sulcus and the normal looking grey matter that wraps that sulcus. You can see similar structures in the other hemisphere.
  - **Right** The black arrow points to a region that **should** be covered by the mask, but isn't. The discolored region is not comparable to anything in the other hemisphere. It is dark like grey matter, but does not wrap a sulcus and is in an area that should be white matter.

- The ventricle is also covered by the mask, but really we want only the enlarged portion of the ventricle to be covered by the mask…not the entire ventricle. We did this on purpose and will fix it later.

ITK-SNAP's semi-automated and manual techniques help make lesion masking faster and more reproducible than the traditional fully manual approach. Here we review a semi-automated approach to segmentation of stroke-related lesions. For more tips on working with large lesions, see the *Stroke Lesions page*.

Why Create a Lesion Mask?

Here are some reasons to mask a lesion:

- You want to measure the volume of the lesion (so you measure the volume of the mask).

- You want to normalize the brain into standard space, so you mask the lesion to indicate that the tissue under the mask should not be considered (because the tissue under the mask is abnormal). This is important for many normalization routines.

- You want to run statistical analyses on lots of lesions to determine how the lesion location and size might affect behavioral measures. Explore Voxel Based Morphometry (VBM) and Voxel Lesion Symptom Mapping (VLSM) to learn more.

Goals

Learn to use ITK-SNAP segmentation software to create an initial lesion mask. In this pipeline we fill the lesion in the left hemisphere without worrying about leaking into ventricles or outside the brain, because we will fix those problems later using FSL.

Data

A dataset, T1w_defaced.nii.gz was kindly provided by A. Kielar. It consists of one native space defaced T1 weighted anatomical image of a patient with a large stroke-related lesion. The data has isotropic voxels which makes it optimal for semi-automatic segmentation in ITK-SNAP. Anatomical MRI data like this must have the face removed in order to deidentify it before posting it publicly. More information about how the data were collected is available here:

Kielar, A., Deschamps, T., Jokel, R., and Meltzer, J. A. (2016). Functional reorganization of language networks for semantics and syntax in chronic stroke: Evidence from MEG. Human Brain Mapping, 37(8), 2869-2893. doi: 10.1002/hbm.23212 PMID:27091757

Setup

First you will need to install ITKSNAP and open T1w_defaced.nii.gz. See *Load and View Images*.

### Select a Region of Interest for Automatic Segmentation

- Make sure the *Crosshair mode* is selected (leftmost icon on *Main Toolbar*). When you move the *3D crosshairs* in one view, they move in the other views as well.

- Position the *3D crosshairs* in the center of the lesion (look at all three views).

- In the *Main Toolbar*, select the *Active Contour Tool* (the snake and staff).

- Using the left mouse button, drag the corners of the active contour selection box, so that the box encompasses the lesion and enlarged ventricles as shown below.

- Use the scroll bars on the right of each image to ensure you have tightly enclosed the entire lesion (including damaged tissue that may surround the CSF). We will worry about excluding the normal extent of the ventricles later.

- Press *Segment 3D* (green box, middle left of figure below) to crop the image to the box.
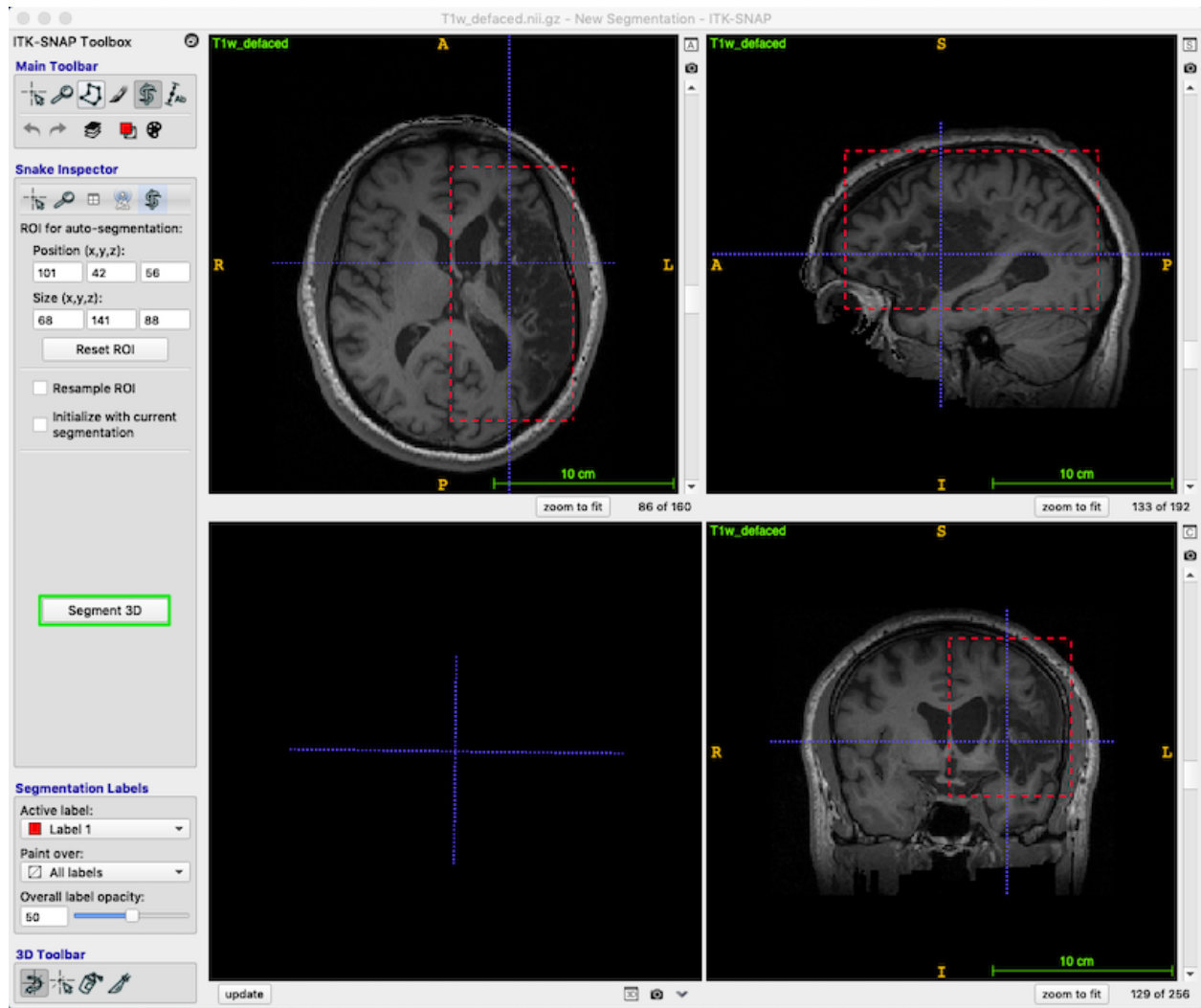
- In the *Segment 3D* panel, ensure that *Current Stage = Step 1/3 Presegmentation Mode*; *Actions = Thresholding* (blue box, upper right in the figure below).

- You have now produced the *speed image*. This is a simplified blue and white segmentation.

- Press the little icon in the right top corner of any of the three slice views (3 tiny green boxes in figure below).
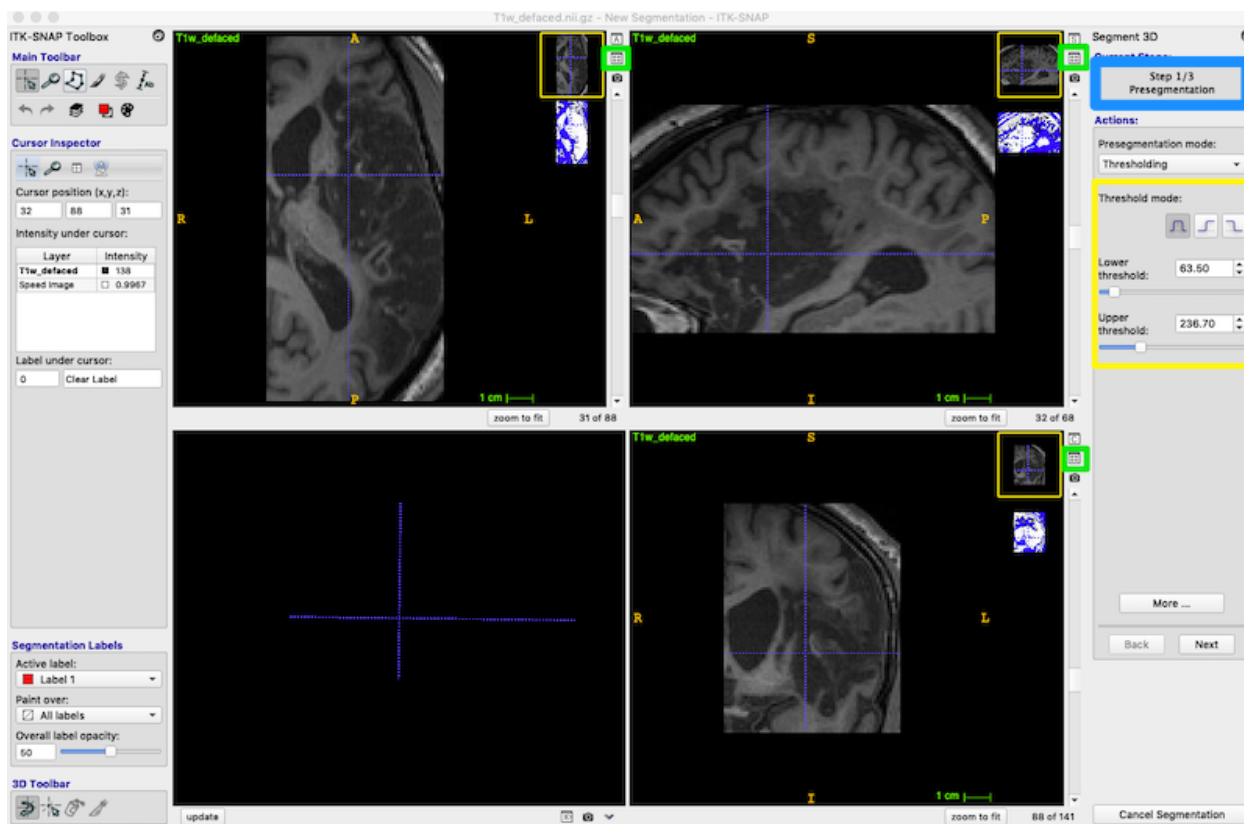
- This icon toggles between the thumbnail view of the speed image (shown above) and the side_by_side view (shown below).

- Side-by-side view is easier to use in the following steps.

- If you want the images to fill the available space, try the *zoom to fit* button (orange box, figure below)
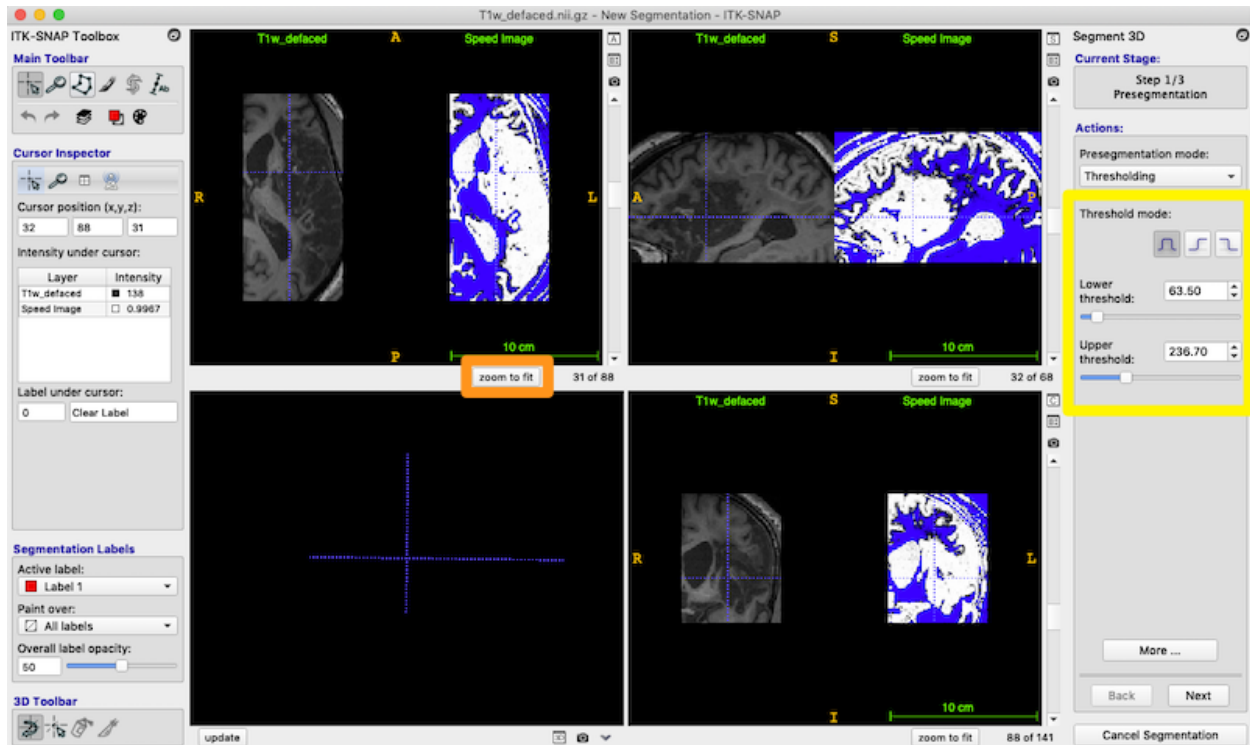
Segment the Lesion

Thresholding to Isolate the Lesion

- Your speed image probably does not look like the one pictured, because the default thresholds are wrong. Now that you are viewing the T1 and speed image side-by-side, you should adjust the thresholds (yellow box on the right).

- Set the lower and uppper thresholds as shown. Your goal is the make the lesioned tissue as white as possible in the speed image, but keep the non-lesioned tissue blue. You should play with the settings.

- white=lesion

- blue=non-lesion

Note: The *Threshold mode* we use is the first of the 3 icons (yellow box, figure above): the complete waveform represents having both an upper and lower threshold. Choose each of the other icons to understand how they control thresholding.

- Press *Next* on the bottom right of the *Segment 3D* panel to continue. (You can go back without losing anything).

- You are now on *Step 2/3*: *Initialization mode*

An Alternative: Clustering to Isolate the Lesion

Thresholding works well when you have a single high quality image, as provided for this tutorial. However, if you have multimodal data (e.g., T1w, T2w, DWI), then you may benefit from using clustering instead of thresholding. The

clustering approach can use converging evidence from all of the images to identify the lesion. Clustering *may* also do a better job than thresholding of identifying regions that are heterogenous and thus not well defined by thresholding.

If you have multiple images, they must have isotropic voxels and be coregistered. See these *useful tools* if you don't have a preferred way to achieve isotropic voxels and coregistration. The video multimodal segmentation in ITK-SNAP provides a good introduction to the clustering approach.

- To use clustering instead of thresholding, choose *Clustering* from the *Presegmentation mode* dropdown box (figure below).
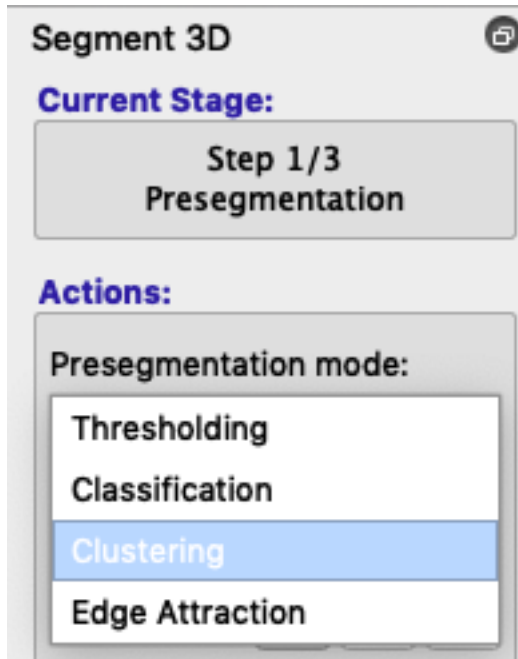
- Instead of **Upper and Lower thresholds**, you will see **Number of clusters** and **Foreground cluster**. By default, the speed image displays the first cluster in the foreground, but you can click the selector next to **Foreground cluster** to see more identified clusters. You can also increase or decrease the number of clusters to discover whether the tissue you want is identified reasonably well. Usually 3-5 clusters will do the job.

You could use clustering even with a single image like this one. Clustering may do a better job of identifying damaged tissue. Keep in mind that you are not limited to doing a single segmentation. You could create multiple segmentations and then combine them.

Set Segmentation Labels

- Segmentation labels are optional for this tutorial because you will only have one segmentation mask open at a time. Nevertheless, it is a good idea to learn how to label different segmentations, so let's do it.

- Follow the instructions in the *labels section* to change *Label 1* to *Lesion*

- Set the *Active label = Lesion* once you have changed the name.

- For the active label, set *Overall label opacity = 50*. This insures 50% opacity, so you can see the anatomical image under the label. You will need to watch how much of the region is filled by the lesion label, so you know when to stop the process.

Place Bubbles

Okay, that was a long detour from the task at hand, but you have learned about clustering and labeling. Now, as you may recall, we are on *Step 2/3*: *Initialization mode*. We are going to use thresholding for the remainder of the lesson.

- In this step, you place bubbles to initialize the active contour.

- First, check that the *Segmentation Labels* are set correctly (red box, figure below). The bubbles will be red and 50% opacity.

- Use the *3D crosshair cursor* to select a location in the middle of the lesion.

- Under *Actions*, press *Add Bubble at Cursor* to insert an initialization bubble (green box, upper right of figure below).

- Place initiation bubbles on various slices to cover examples of both damaged tissue and the heterogeneous CSF-filled portion of the lesion. The figure below shows numerous bubbles on the image and in the *Active bubbles* table on right of the figure.

- In the figure below, I have put a yellow box around one of the bubbles. This bubble was not placed very carefully, and you can see that it overlaps good tissue next to the ventricle. Don't worry! This is fine. When we get to the next step the bubbles will morph to fill the lesioned tissue and shrink away from the healthy tissue.

- Scroll through each of the three orthogonal views to ensure you have a good selection of bubbles. You can delete bubbles or change their sizes from the *Active bubbles* table on the right.

Note: As you scroll through the slices, the bubbles appear to change size. This is because they are spheres, so they appear smaller as you move away from their centers.
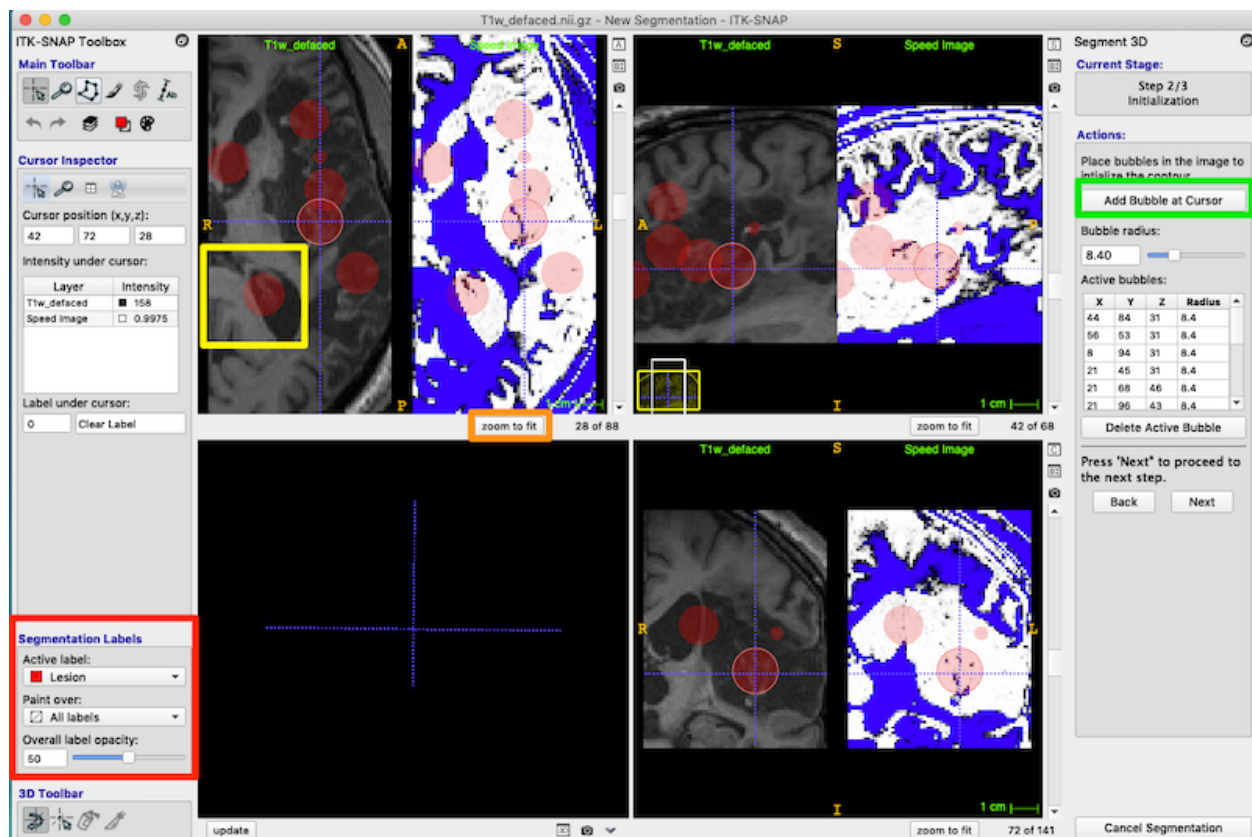
- Press *Next* in the *Segment 3D* panel on the right to continue. You can always go back without losing your work.

- You are now on *Step 3/3 Evolution*.

Run Active Contour Segmentation

- In *Step 3/3 Evolution*, you grow the bubbles to fill the lesion.

- Press *Set Parameters* (red box on right of figure below) to bring up the *Active Contour Evolution Parameters* window shown below.
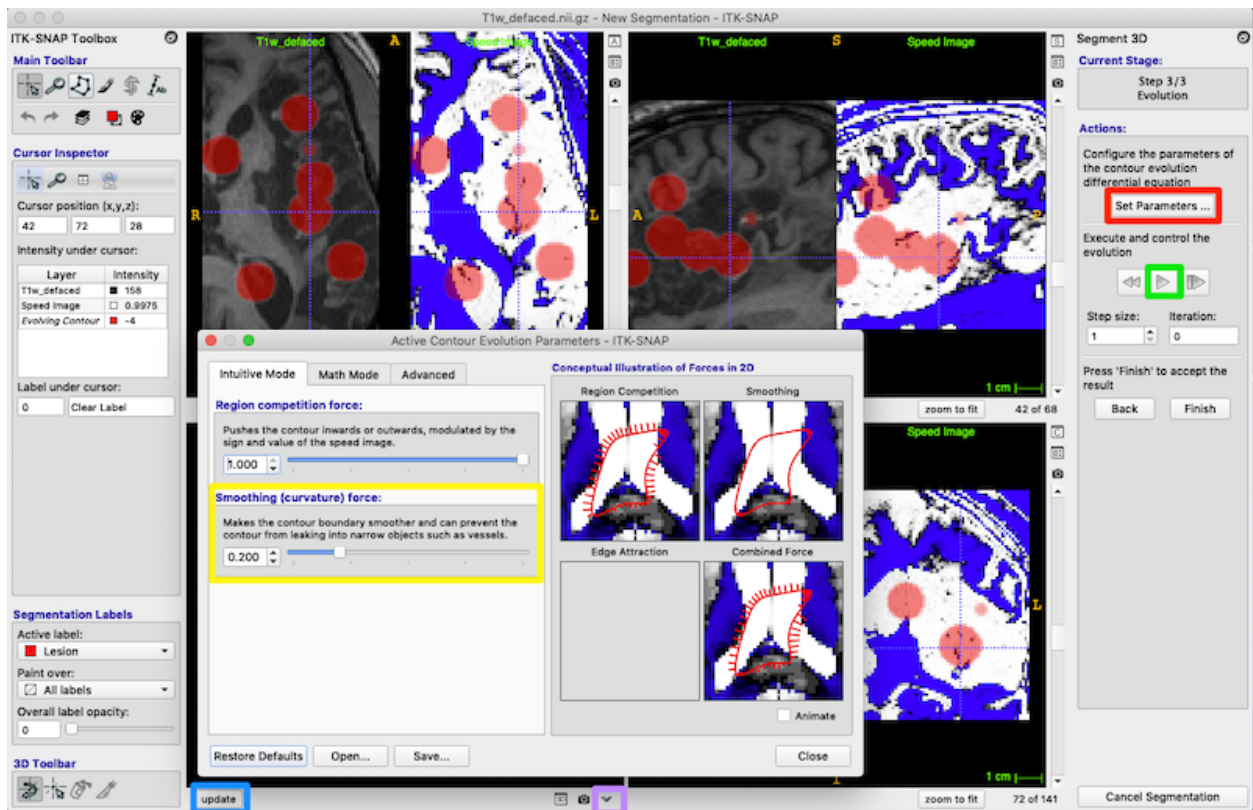
- Set *Smoothing (curvature) force* to *0.8* (yellow box, figure above) and press *Close* on the *Active Contour Evolution Parameters* window. This will prevent the bubbles from growing long spindly fingers, which is good because we want to discourage the bubbles from growing into every sulcus they encounter.

- To see your bubbles in 3D, press the *update* button under the 3D panel (blue box, lower left of figure above).

- Alternatively, select the down arrow in the bottom center of the figure below (tiny purple box, figure above). You can choose *Continuous Update* to watch the 3D shape emerge as the contour evolves.

- Under *Actions*, press the play button (green box on middle right) to start evolution. It changes into a pause button. After about 950 iterations, press the pause button to stop the evolution.

- You can evolve the contour one iteration at a time with the button to the right of the play/pause button (purple box, middle right of figure below).

- If you are not using *Continuous Update*, then as the contour evolves, you can pause the evolution and update the 3D view. Again press the *update* button under the 3D panel to see the 3D shape.

- If you are not satisfied with the segmentation, you can return to the original unevolved bubbles with the button on the left of the play/pause button (red box on middle right of the figure below).

- Exit semi-automatic segmentation mode: On the right, under *Actions*, press the *Finish* button (yellow box, on the right in the figure below). You can always click *update* to get the latest 3D rendering of the lesion structure.
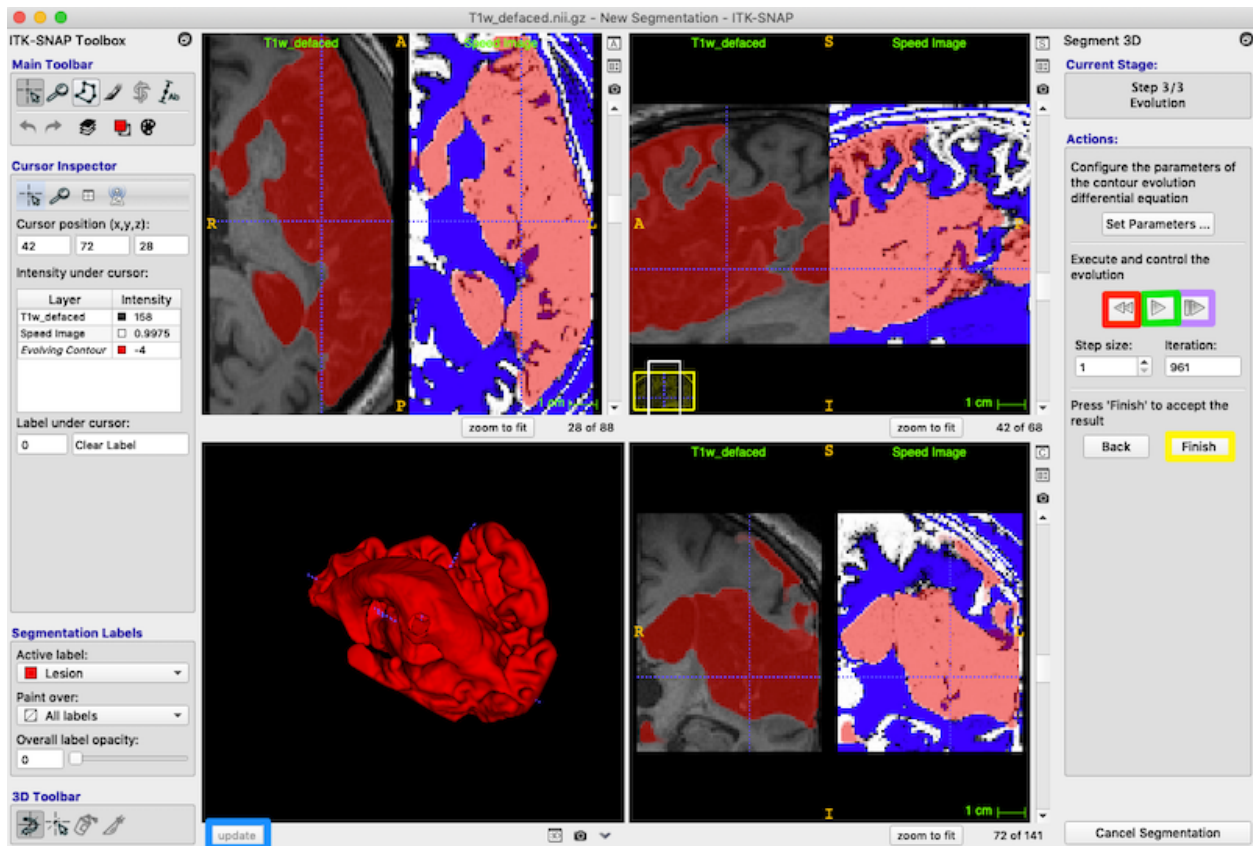
Note: In the 3D panel, use the right-mouse to change zoom level on the 3D volume, and the left-mouse to rotate the 3D volume. You can also move the 3D volume by moving the sliders on the other three panels.

Save Workspace

Before editing the segmentation, let's save our work as a workspace in ITK-SNAP. This will include saving the segmentation mask we just created.

- From the ITK-SNAP menu, select *Workspace → Save Workspace As*
- ITK-SNAP complains that the segmentation image does not have a filename.
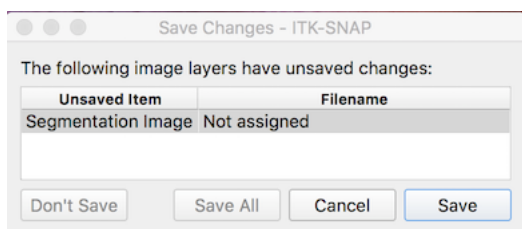
- Press *Save* and choose a name and location for your segmentation file (e.g., *sub-001_lesion_mask.nii.gz*).

- You will now be asked to save the workspace. Choose a workspace name and location as well (e.g., *sub-001.itksnap* in the same directory as the segmentation).

- Now you can easily load the anatomical image and any segmentations by choosing the workspace from the menu.

Editing the Segmentation

Now we want to evaluate and edit our work. We can do some of this with ITK-SNAP, but FSL also offers some useful tools. The lesion may have leaked into neighboring ventricles or even outside of the brain. Both 2D and 3D editing tools are available in ITK-SNAP. See *2D segmentation* for more information about using these editing tools.

You now have experience with semi-automated and manual segmentation tools in ITK-SNAP. Next, I introduce you to additional tools that can be used to clean up the mask you created. Most of these tools depend on FSL and the *UNIX command line*. So this is a much deeper dive into the NIFTI image format and associated thorny problems. If you don't have any experience with the command line, this might be a good time to take a break. If you are comfortable at the UNIX command line and already have FSL installed, then you will want to download some additional scripts as described below, and put them in your path.

Continue to the section *Create a Brain Mask*.

SPM12 Revised Normalization

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Author: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2018_11_17
Date Updated: 2019_06_26
Tags: SPM, normalization, standard space, lesion, anat
Software: At least Matlab 2016B, SPM12

Goal: Warp a native space anatomical image into standard MNI space. As established by the *lesion normalization experiments*, SPM12 does a nice job of normalizing the head, even the lesioned head, without a lesion mask. It is beneficial to normalize the head **before** drawing the lesion for the following reasons:

- We can visually inspect the head, insuring that the normalization was good.

- The head will be correctly aligned, with a nice straight midline. This facilitates comparing one hemisphere to the other which is often very helpful in identifying which tissue is abnormal.

- The head's appearance will be more consistent with all other heads the tracer encounters, which should facilitate consistency (yet to be determined)

- We reduce the total number of steps, because we no longer need to normalize the lesion.

- By reducing the number of steps, we reduce the possibility of interpolation and thus avoid introducing non-binary values into the mask.

- The standard brain mask can be used to prevent auto segmentation of the lesion from leaking outside the brain.

> **Warning:** This *normalization first* approach works well for homogenous high-resolution (e.g., ~1.0mmx1.0mmx1.0mm) research T1w images. However, the heterogenous datasets that sometimes characterize stroke datasets (CTs, clinical scans) are not necessarily well suited to this approach.

The images below demonstrate the consistency and corrected alignment of the normalized heads (left=native; right=normalized). Data were provided by *Liew et al. 2018*.
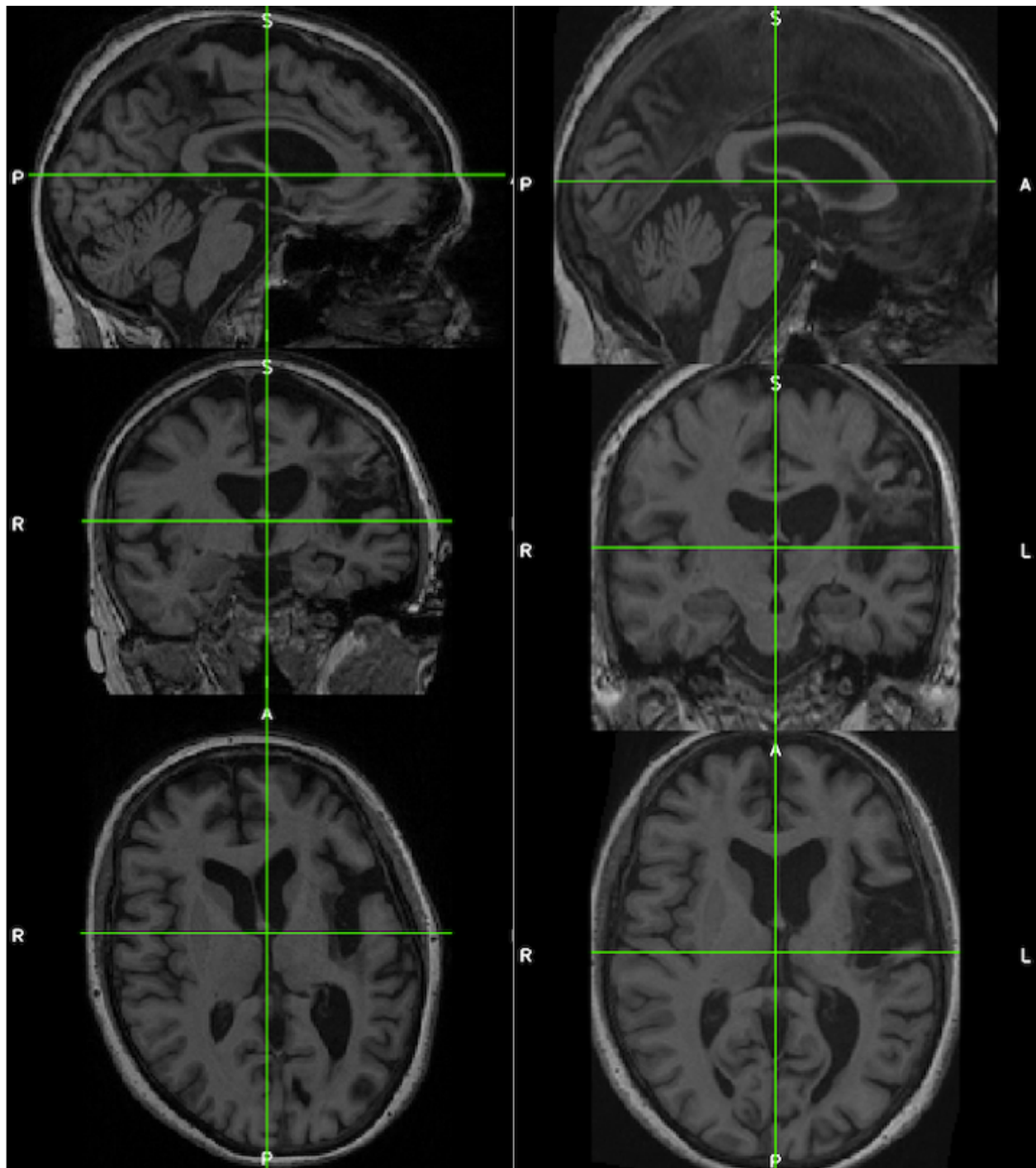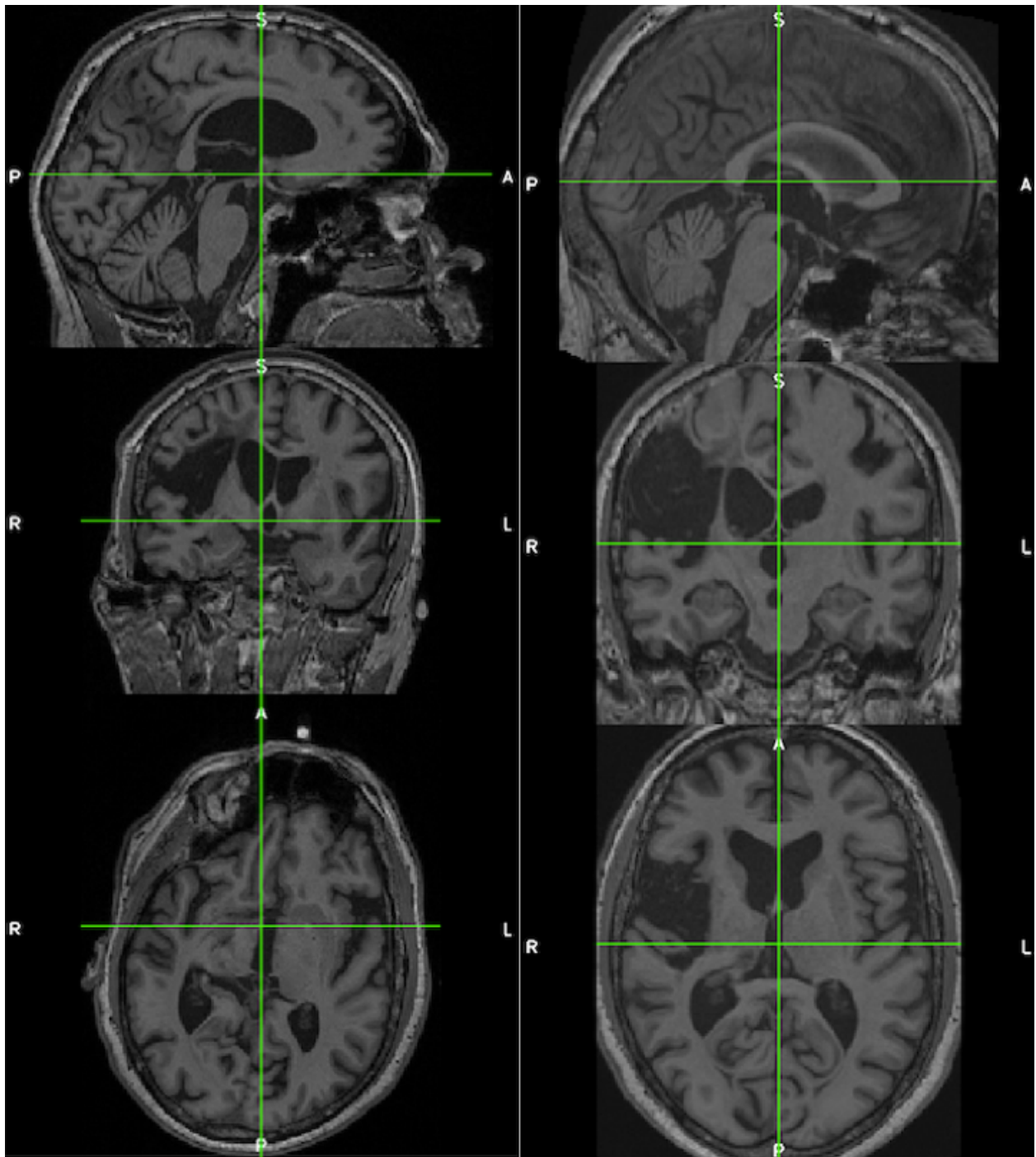
See the suggested lesion drawing approach described in the introduction to *ITK-SNAP Segmentation of Stroke-related Lesions*.

Data

The anatomical image must **not** be zipped (i.e., *T1w.nii* is okay, but *T1w.nii.gz* is not). *Sample data* is described, and available for download.

Steps

Segment: Estimate Warp Field

- Open Matlab (and optionally navigate to the relevant directory).

- Type *spm* at the Matlab command prompt. This will open spm

- Click the *fMRI* button which will open the windows you need.

- Select *Batch* and from the menubar: *SPM Spatial Segment*
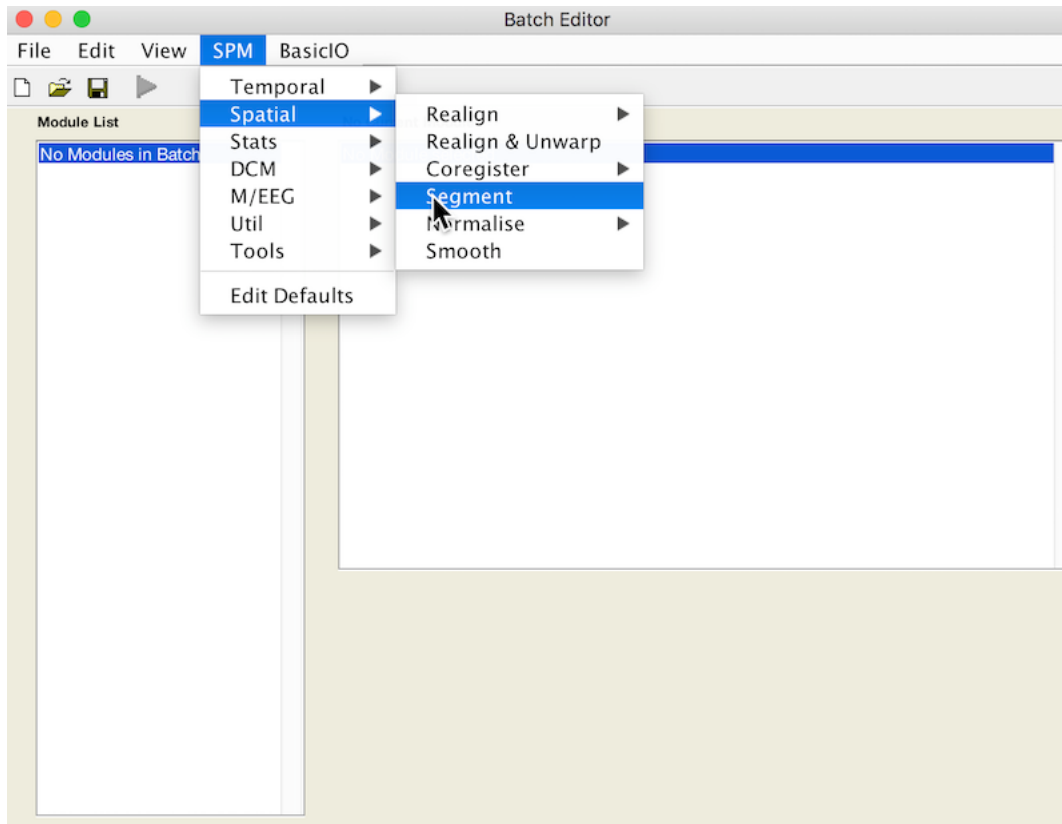
- Under *Volumes*, an **X** is displayed, indicating that this parameter must be specified. Specify the native space anatomical image (whole head)
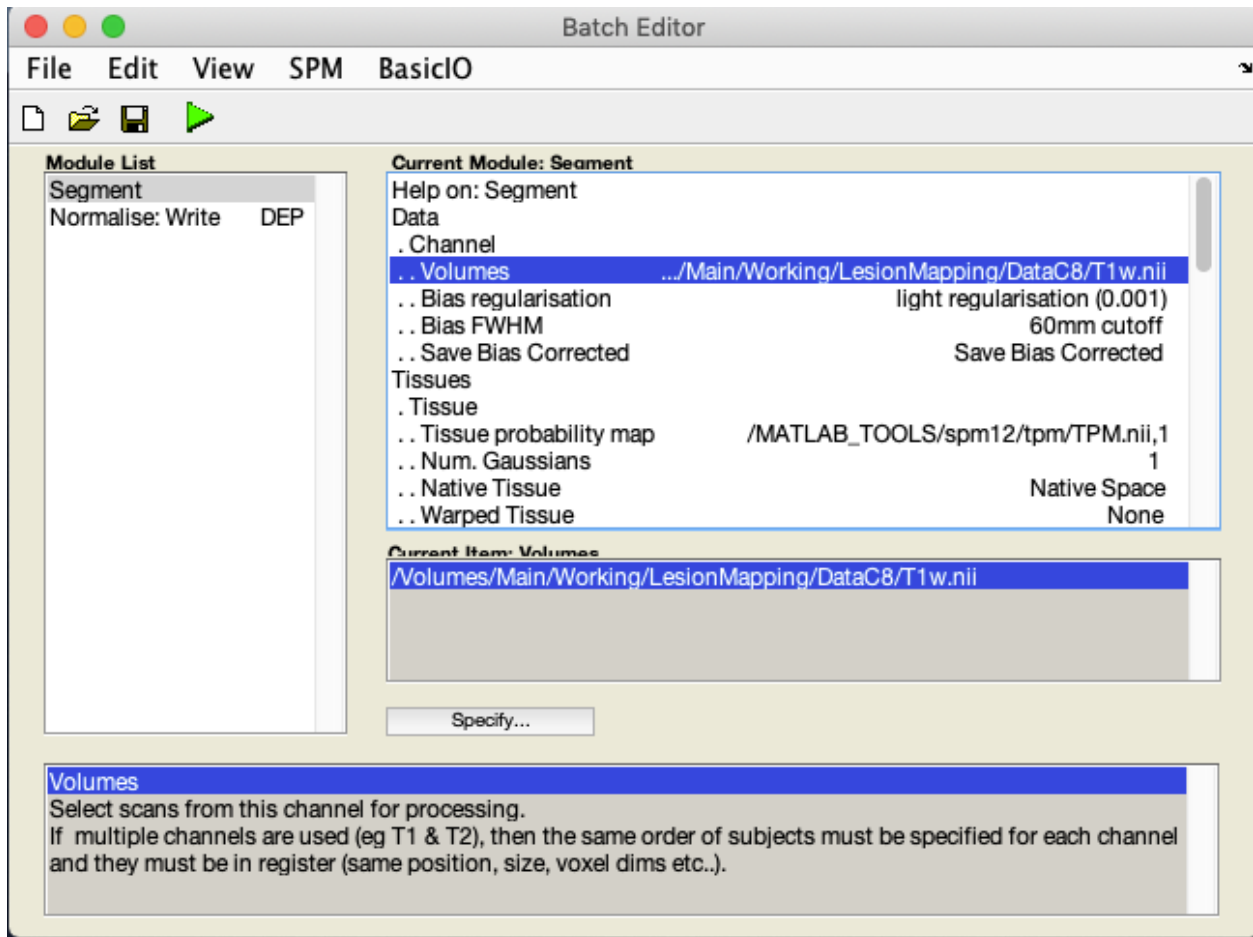
- A few lines below *Volumes* you will see *Save Bias Corrected.* By default it is set to *Save Nothing.* Select it and choose *Save Bias Corrected*

- Scroll down to the bottom of the module options, and set *Deformation Fields* to *Forward.* The *Forward* warp will be used to warp files from native space into standard space. If you elect to create an Inverse deformation as well, choose *Inverse + Forward.* That would create an additional warp allowing you to deform files from standard space into native space.

- This is your first batch module. Now we'll add another.

Normalise: Apply Warp Field to Bias Corrected Head

1. From the *Batch* editor, choose: *SPM Spatial Normalise Normalise: Write*

2. This adds a second module to your batch

3. Double click *Data. Deformation Field* and *Image to Write* now appear in the menu.

4. For *Deformation Field*, click *Dependency* and select *Segment: Forward Deformation* to specify the **y** file, even though you have yet to create that file. Click OK.
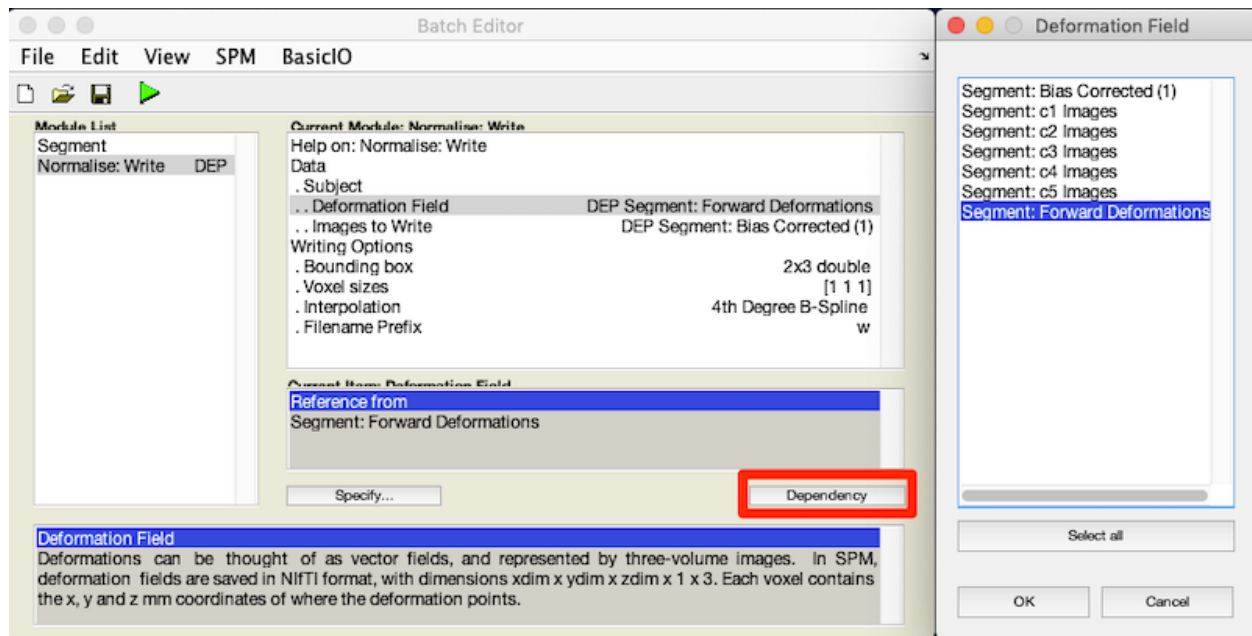
- For *Image to Write*, click *Dependency* and select Segment: Bias Corrected (1)

  - By choosing to bias-correct we correct for greyscale gradients that could result in misleading tissue intensities across the image.

- Under *Voxel sizes* enter *[1 1 1]*

  - By choosing *[1 1 1]* instead of *[2 2 2]*, we are retaining the higher resolution and tissue contrast of the original image. This assumes the original image is higher resolution.

Note: There are 8 1x1x1 voxels in each 2x2x2 voxel, so the resolution difference is considerable.

Run Batch and View Results

- Run the batch module by clicking the green go button (upper left on batch editor).
- Segmentation produces:

  - Files prefixed with **c** for each tissue type and the skull.
  - A *seg8.mat* file containing segmentation parameters.
  - A deformation field image with **y** prepended.
  - A bias corrected T1w image with **m** prepended.

- Normalization produces:

  - An anatomical image in standard MNI space, also with a **wm** prepended.

8.2 Working with Clinical Data

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu

Date Created: 2019_06_25

Date Updated: 2019_08_17

Tags: anat, lesion, Image Fusion, CT, Film

Clinical data differs from our typical research MRI data in several ways:

- Voxels are often extremely anisotropic. In clinical scenarios patients may be unable to hold still for the long periods routinely expected for research. Clinicians meet this challenge by collecting multiple short orthogonal scans that have high in-plane resolution but very thick slices.

- Clinical scans are more likely to involve a contrast agent.

- Clinical scans are more likely to be CT instead of MRI (CT is fast, and safe for people with metallic artifacts).

8.2.1 Simple Image Fusion

As noted above, clinical scans are often characterized by separate images in each of the three orthogonal planes. These images have high inplane resolution but extremely thick slices. It is possible to combine these images in a process called *fusion*. At its most sophisticated, fusion involves the use of clever algorithms that optimize features of interest. Here I describe the fusion process using averaging. This can easily be accomplished in FSL tools. Download and unzip the following data: fusion.zip. cd to the fusion directory to find three images:

```
T1w_defaced_axial.nii.gz
T1w_defaced_coronal.nii.gz
T1w_defaced_sagittal.nii.gz
```

Find the smallest inplane voxel size in all three images, and reslice all three of the images to isotropic voxels using this highest resolution value. In the fusion.zip files, that is 0.72mm in T1w_defaced_axial.nii.gz You can use iso.sh described above in the section on *correcting anisotropy*. Here are the three commands you need to type:

```
iso.sh T1w_defaced_axial 0.72
iso.sh T1w_defaced_coronal 0.72
iso.sh T1w_defaced_sagittal 0.72
```

Register all of the high resolution isotropic images together using reslice.sh. First, register the coronal image to the axial image:

```
reslice.sh T1w_defaced_coronal_0.72mm T1w_defaced_axial_0.72mm
```

Also register the sagittal image to the axial image (Note these are slow commands to run, you now have very high resolution images to register):

```
reslice.sh T1w_defaced_sagittal_0.72mm T1w_defaced_axial_0.72mm
```

Now, average the images together:

```
fslmaths T1w_defaced_axial_0.72mm.nii.gz -add T1w_defaced_coronal_0.72mm_resliced.nii.
→gz -add T1w_defaced_sagittal_0.72mm_resliced.nii.gz -div 3 mean_T1w
```

The above command adds each of the registered images together and then divides the results by 3 to get the mean_T1w.nii.gz image.

8.2.2 CT Scans

Gantry Tilt

One problem encountered in CT images is the elongation of the skull resulting from gantry tilt as in the image on the left below. If you use dcm2niix for conversion from DICOM to NIFTI, an image with gantry tilt correction will be generated automatically (image on the right, below).
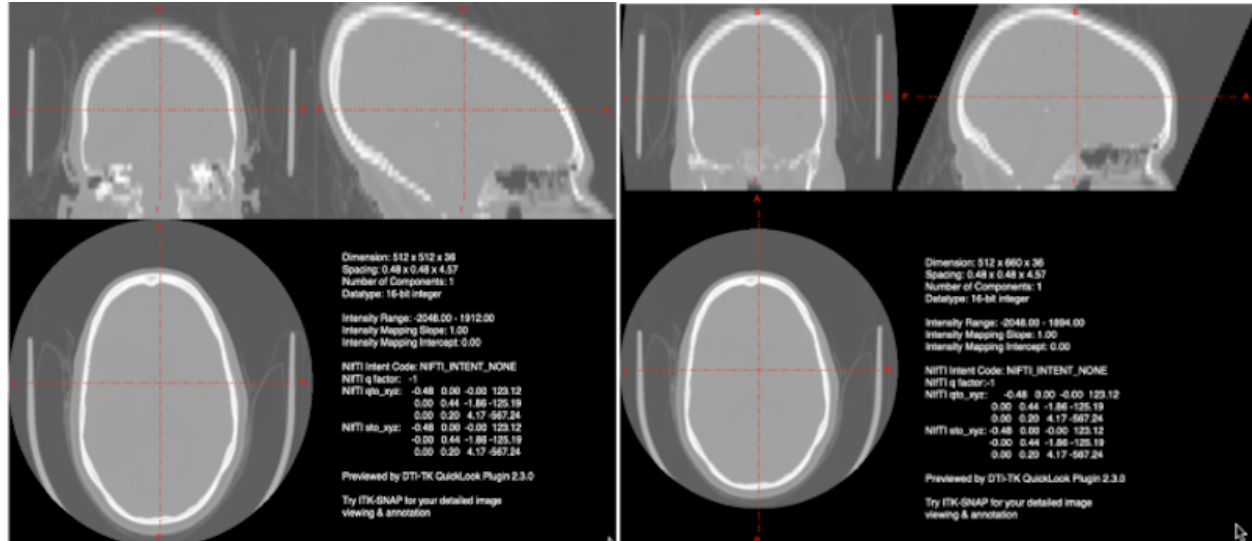

Fig. 1: **Left** Gantry tilt causes elongation of the skull in the nifti file. **Right**: dcm2niix corrects for the gantry tilt in an image named with the keyword `Tilt`.

Note: This tilt correction changes the dimensions of the image! See the *JPEG Stack Export section* for a description of how to reslice your distorted image into undistorted space. It is the last step in producing a good CT nifti image.

CT Windowing (Contrast) Issues

Whatever tool you use for conversion of CT DICOMS (dcm2niix MRIconvert etc.), you are likely to see windowing problems in the NIFTI format. In the picture below, we display the DICOM image on the left and the NIFTI image on the right. Obviously the NIFTI image is not suitable for lesion drawing.
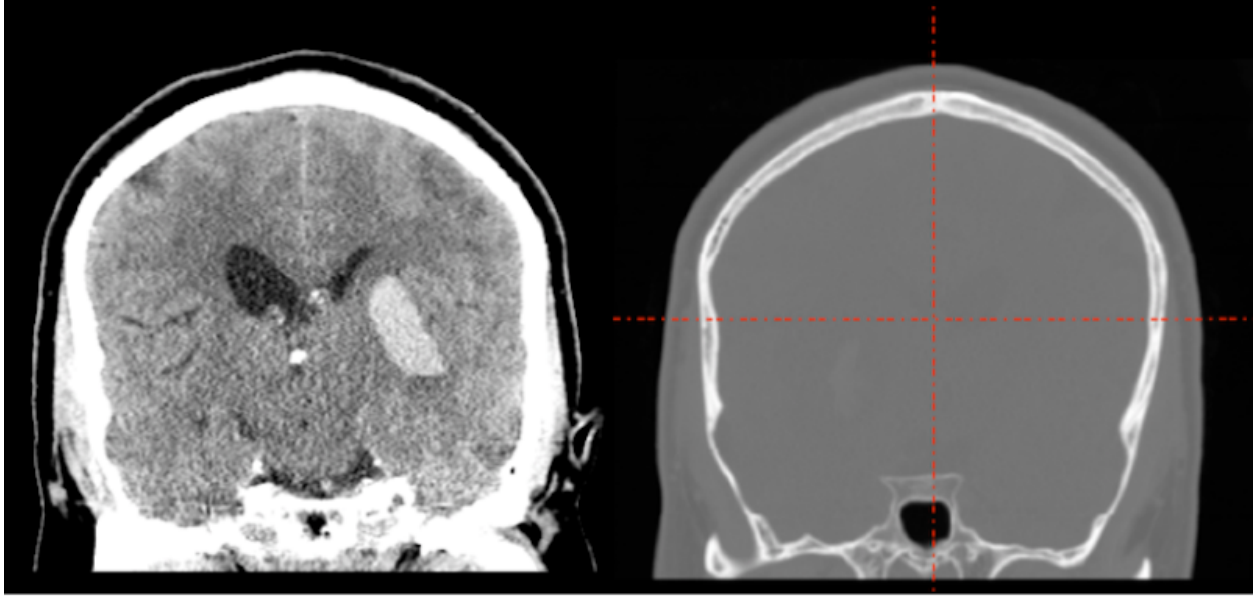
What happened? The CT image contains large negative numbers in the dark area surrounding the head. The bone of the skull is extremely bright. The soft tissue contrasts are small compared to the background and bone values.

There are a couple of partial solutions: A script scaleCT.sh, which I wrote to call FSL utilities to correct the problem; and a solution which is part of the SPM12 Clinical toolbox by Chris Rorden (This toolbox is specifically designed to normalize CT and other clinical scans using lesion weighting. In the process of creating the normalized data, though, it generates a new native space CT with the prefix c. This new image has much better contrast than the original but still is not great for drawing lesions).

However, both of these windowing solutions obscure features of the image which are visible in the original dicoms. If you are willing to spend the time, then the *JPEG Stack Export* described below preserves the original dicom detail. This requires that you install FIJI and the NifTio plugin for FIJI.

JPEG Stack Export

Tools

- Horos (to export DICOMS to JPG)

- FIJI to get the image sequence and export it to NIFTI

- FSL 6.0.1 or greater: You want *fslcpgeom* and *flirt*.

Steps when you have DICOMS

1. *Horos File Export* (JPG)

2. FIJI: *File Import Image Sequence*: point at jpg dir, they are viewable now.

3. FIJI: To export properly to NIFTI, you want to orient the image correctly in FIJI. My current experience suggests you want the head to appear upside down in FIJI so that it will be rightside up in NIFTI (this is true for the axial, coronal and sagittal). In addition, for the sagittal image, you want the left hemisphere to appear at the beginning of the stack (that is, when the FIJI slider is on the left).

4. FIJI: *Image Transform Flip Vertically* (Yes, all the slices): This should make the image upside down).

---

5. FIJI: Sagittal only: *Image Transform Flip Z* (Yes, all the slices): This should switch the through plane (Left-Right).

6. FIJI: File save as Nifti1.

7. Create NIFTI from DICOMS: To get the correct labels and dimensions applied to the JPEG stack image, you will need the NIFTI file created by converting DICOMS to NIFTI. Generally dcm2niix is quite nice for this, especially since it detects gantry tilt. However, MRIconvert is less likely to decide that your CT images should be split into several smaller images.

- fslcpgeom:

    - **Bad_contrast**: The NIFTI file created from the DICOMS should have the correct labels and dimensions, but terrible contrast (this is NOT the gantry tilt corrercted image).

    - **Bad_dimensions**: The NIFTI file created from the JPEG stack, has good contrast, but the dimensions are all wrong and labels are missing.

    - As long as *Bad_contrast* and *Bad_dimensions* are in the same orientation (when you view them), you can copy the geometry of *Bad_contrast* to *Bad_dimensions* to produce a NIFTI file with good contrast and good dimensions:

        ```
        fslcpgeom  Bad_contrast Bad_dimensions
        ```

    - Do NOT use the gantry tilt corrected image for fslcpgeom.

    - **Check your output**: Clinical CTs appear to be very heterogeneous. Make sure the labels on the anatomy are correct (i.e., S [superior] is at the top of the head, A [anterior] is at the front of the head etc.

- Ensure your images are in the standard orientation. This will make everything easier. Especially if you are going to apply tilt correction, make sure the tilt corrected image has also undergone fslreorient2std:

    ```
    fslreorient2std good_image good_image_reoriented
    fslreorinet2std tilt_corrected_image tilt_corrected_image_reoriented
    ```

- Once you have your good_image_reoriented, you still need to apply tilt correction. You can do this with reslice.sh

    ```
    reslice.sh good_image_reoriented tilt_corrected_image_reoriented
    ```

Film JPEG NIFTI

Hopefully you'll never have to do this. It isn't optimal, but seems to work.

- Look at your film. It should contain information about slice position in mm. From consecutive slices you can calculate slice thickness. In addition, you should have a resolution (e.g. 256 x 256) and FOV (field of view) (e.g., 230 mm). From these items you can determine in-plane voxel size (230/256=0.898). You will need this information later.

- You need digital images of the film. I used an ipad with the white flashlight app as a light table, then took square pictures of each panel in the film. I tried hard to keep everything level and to ensure that each panel filled the square picture frame.

- Even if you do a really good job of keeping each panel straight in your photo, it seems like whoever creates film does not have in mind that someone might do this. That means the various slices you get aren't properly aligned.

- These pictures may need to be deidentified. I used a photo editing program and blacked out the patient name and institution.

- You may want to resize the images: my originals were 3024x3024. I resized them to 512x512. At 512x512, pixel dimensions will be half my original calculation (e.g., 0.449 mm in-plane).

- As described above, we'll now use FIJI *File Import Image Sequence*: point at jpg dir. You need to ensure that these images are grayscale. My phone takes color images by default and NIFTI cannot handle that. When you import the image sequence, FIJI gives you the opportunity to *Convert to 8 bit Grayscale*, which does the trick.

- As previously mentioned the slices are unlikely to be properly aligned.

- FIJI: To see this, choose *Image Scale*, and set your voxels to the correct sizes (e.g. 0.449, 0.449 and 6.5 in my case). FIJI generates a second image.

- FIJI: Choose *Image Stacks Orthogonal Views*. My original image was sagittal. On the left, below, you see the uncorrected coronal view. Red arrows point to the locations on the skull that are really out of alignment.
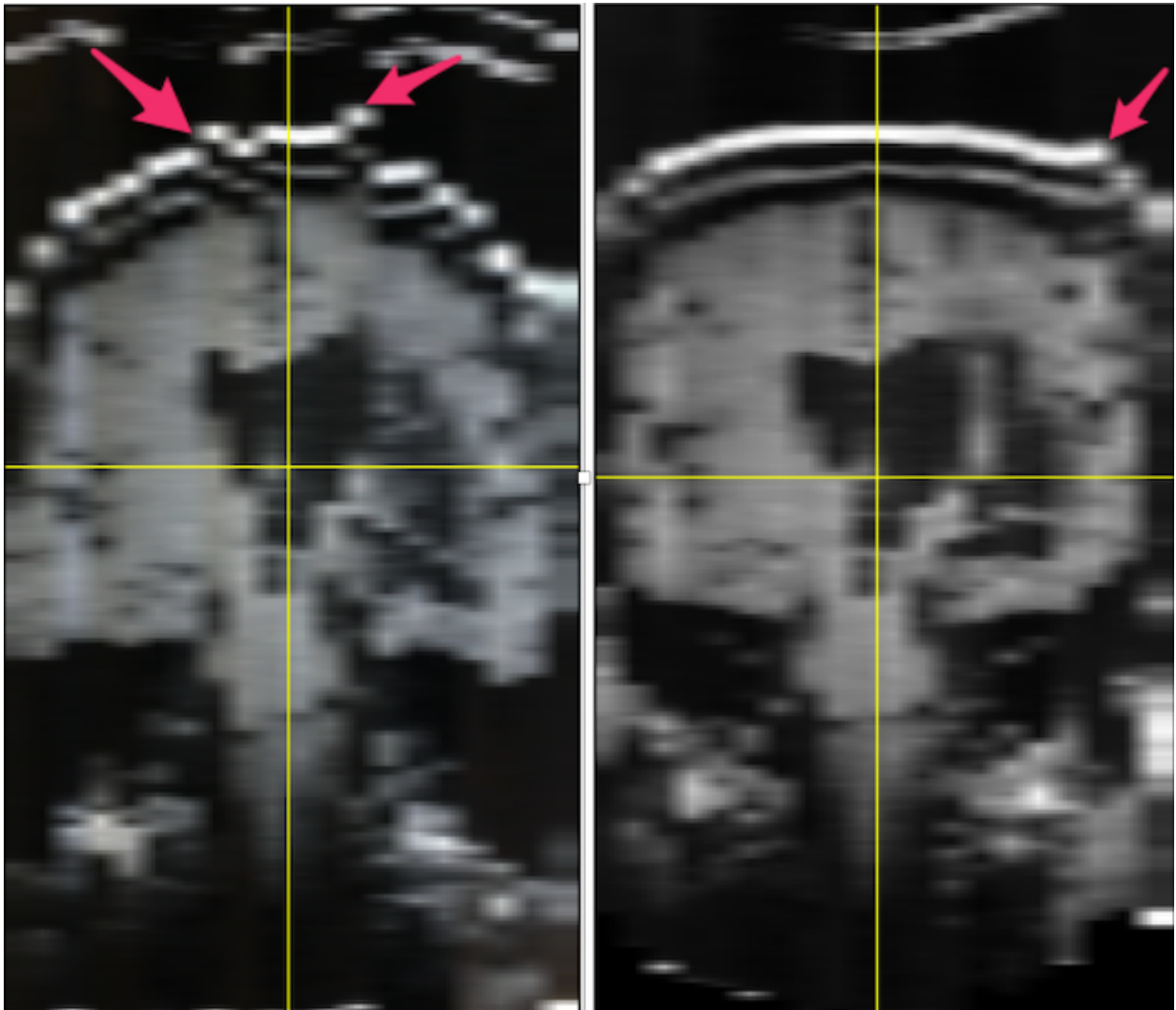
Fig. 2: **Left**: MR images from film: when stacked, the slices do not line up (illustrated by the red arrows pointing to misaligned skull. **Right**: The alignment of the stacks has been improved using FIJI plugins stackreg and turboreg.

- FIJI has many plugins available. I chose and installed stackreg and its dependency turboreg.

- Using my original unmodified stack (which may or may not be important), I selected the middle slice as the anchor and ran stackreg by selecting it from the plugins menu. I kept the default rigid body registration to avoid

any possibility of altering the lesion. The result can be seen in the image above on the right. It is not perfect (see the arrow), but it is considerably improved compared to the image on the left.

- FIJI: flip vertical, flip z. Save as NIFTI1: anat1

- `fslswapdim anat1 -z -x y anat2` put anat1 into the standard orientation, but without anatomical labels. You will have to play with these options.

- Unlike the *Steps when you have DICOMS*, I did not have a NIFTI image for this subject to copy my geometry from. Fortunately, I had an image from another subject with the same dimensions, and this worked well: `fslcpgeom sub-0304_reoriented.nii.gz anat2.nii.gz`

8.3 Lesion Normalization

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2019_06_25
Date Updated: 2019_11_14
Tags: anat, lesion, normalization

8.3.1 ANTS

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2018_08_10
Date Updated: 2019_09_18
Tags: ANTS, normalization, standard space, lesion, anat
Software: ANTS_2, and probably FSL and optiBET.sh

Data

- Images can be zipped or not (i.e., *T1w.nii* and *T1w.nii.gz* are OK).

- A lesion mask with values of 1 in the lesion and 0 elsewhere.

- A template image. In this script we assume the template image is in the FSL directory here: /usr/local/fsl/data/standard/MNI152_T1_2mm_brain.nii.gz.

- A skull-stripped structural (anatomical) T1 weighted image. See *Skull stripping with optiBET*

Steps

- **Goal**: Warp a native space lesion mask (and native space anatomical image) into standard MNI space.

- **Requirements**:

  - ANTS installed and working.

  - FSL installed and working.

    * FSL provides the template image (MNI152_T1_2mm_brain.nii.gz)

    * optiBET.sh relies on FSL.

- Place the ant_reg2.sh script in your path (left-click to view script; right-click to download).

The script

ant_reg2.sh assumes we are in the directory with the images we need. It further sets the subject variable ${sub} to be the basename of the directory (you can change this). ant_reg2.sh expects 2 arguments:

- a T1 anatomical image (skull stripped–optibet.sh suggested)

- a lesion mask (1=lesion; 0=non-lesion)

- e.g. ant_reg2.sh sub-001.nii.gz sub-001_LesionSmooth.nii.gz

The main tools called in **ant_reg2.sh** are antsRegistration and antsApplyTransforms. These replace ANTs and WarpImageMultiTransform respectively. ANTs and WarpImageMultiTransform were called in the old ANTS. Registration is so complex that there are a couple of scripts to implement it, a quick one, antsRegistrationSyNQuick.sh, and a slower one, antsRegistrationSyN.sh. ant_reg2.sh first creates the inverse lesion mask using the ImageMath function, then calls antsRegistrationSyN.sh with that masking. After creating the affine and warp registrations, they are applied in antsApplyTransforms. The approach follows the basic brain mapping example.

Alternative approaches

- **ant1** For comparison we ran the old ants tools, ANTs and WarpImageMultiTransform, without source weighting. See ant_reg1.sh

- **ant3** We also ran the ant2 procedure described using a different brain mask. We used the final result of fsl_anat_alt instead of the result of optibet. This was motivated by sometimes odd looking brain masks produced by optibet (at least one was clearly wrong). Other than the starting mask, the script was the same as the ant2.sh script. See ant_reg3.sh.

8.3.2 FSL Lesion Normalization

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2018_07_27
Date Updated: 2019_07_30
Tags: FSL, normalization, standard space, lesion, anat
Software: FSL 5.10 (or similar), The zip file of scripts described below

- **Goal**: Warp a native space T1 weighted image and lesion mask into standard (MNI) space using FSL.

- **Requirements**: You need to have FSL installed and working. Place the three scripts described below (*lesion_norm_fsl.sh*, *fsl_anat_alt.sh*, and *optiBET.sh*) in your path. Some familiarity with Unix command line processing is assumed. Once everything is in your path, you will run *lesion_norm_fsl.sh* as illustrated below.

Data

- Images can be zipped or not (i.e., *T1w.nii* and *T1w.nii.gz* are OK).

- A structural (anatomical) T1 weighted image.

- A lesion mask with values of 1 in the lesion and 0 elsewhere.

The 3 Scripts

lesion_norm_fsl.sh assumes we are in the directory with the images we need. It expects 2 arguments:

- a T1 anatomical image

- a lesion mask (1=lesion; 0=non-lesion)

e.g.:

```
lesion_norm_fsl.sh sub-001.nii.gz sub-001_LesionSmooth.nii.gz
```

lesion_norm_fsl.sh calls the two scripts described below (*fsl_anat_alt.sh* and *optiBET.sh*)

fsl_anat_alt.sh is an altered version of *fsl_anat* that does several things:

- It uncomments the lesion mask handling calls that *fsl_anat* was not using.

- It calls *optiBET.sh* to improve the quality of BET, especially for lesioned brains.

- It assumes that because *optiBET* is so much better, we can use the skull-stripped brain for linear registration.

optiBET.sh is a simplified version of the original *optiBET*. It calls *bin/bash* instead of *bin/sh*, and does the **skull stripping** described below using the FSL options.

Why Skull Strip with optiBET?

- Registration of the participant brain to the standard brain is generally better accomplished if both brains are skull stripped (but see *SPM12*). Skull stripping is difficult, especially for individuals with large lesions.

- The *optiBET.sh* script is impressive, but you should always look at the skull stripping results and edit them if they are bad. Once you have a good skull strip (all the brain, and nothing but the brain), you are ready to go to the next step in your processing journey.

The Originals

- The original **fsl_anat** is described here: fsl_anat

- The original **optiBET** is described here: optiBET

8.3.3 SPM Lesion Normalization

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Author: Alyssa Sachs
Date Created: 2018_06_21
Date Updated: 2019_11_14
Tags: SPM, normalization, standard space, lesion, anat
Software: At least Matlab 2016B, SPM12

Goal: Warp a native space lesion mask (and optionally a native space anatomical image) into standard MNI space.

Data

- Images must **not** be zipped (i.e., *T1w.nii* is okay, but *T1w.nii.gz* is not).

- A structural (anatomical) image, usually T1 weighted, but could be T2 or flair weighted. All testing here was done with T1 weighted images.

- A lesion mask with values of 1 in the lesion and 0 elsewhere.

- To facilitate script processing (see *Batch and Scripting*), it is useful for equivalent files in different subject directories to have equivalent names (e.g. T1w.nii in all directories rather than sub-030_T1w.nii, sub-031_T1w.nii etc.).

- You can use symbolic links to retain your original naming structure while facilitating the scripting, e.g., `ln -s sub-030_T1w.nii T1w.nii`. Just insure that in the instructions below, you select the link, so you are operating with simplified file names. N.B. aliases are not the same as symbolic links.

Note: There must be a way to handle more complex naming. I'll update this if I ever figure it out, or some kind soul explains.

Steps

Segment: Estimate Warp Field

- Open Matlab (and optionally navigate to the relevant directory).

- Type *spm* at the Matlab command prompt. This will open spm

- Click the *fMRI* button which will open the windows you need.

- Select *Batch* and from the menubar: *SPM Spatial Segment*
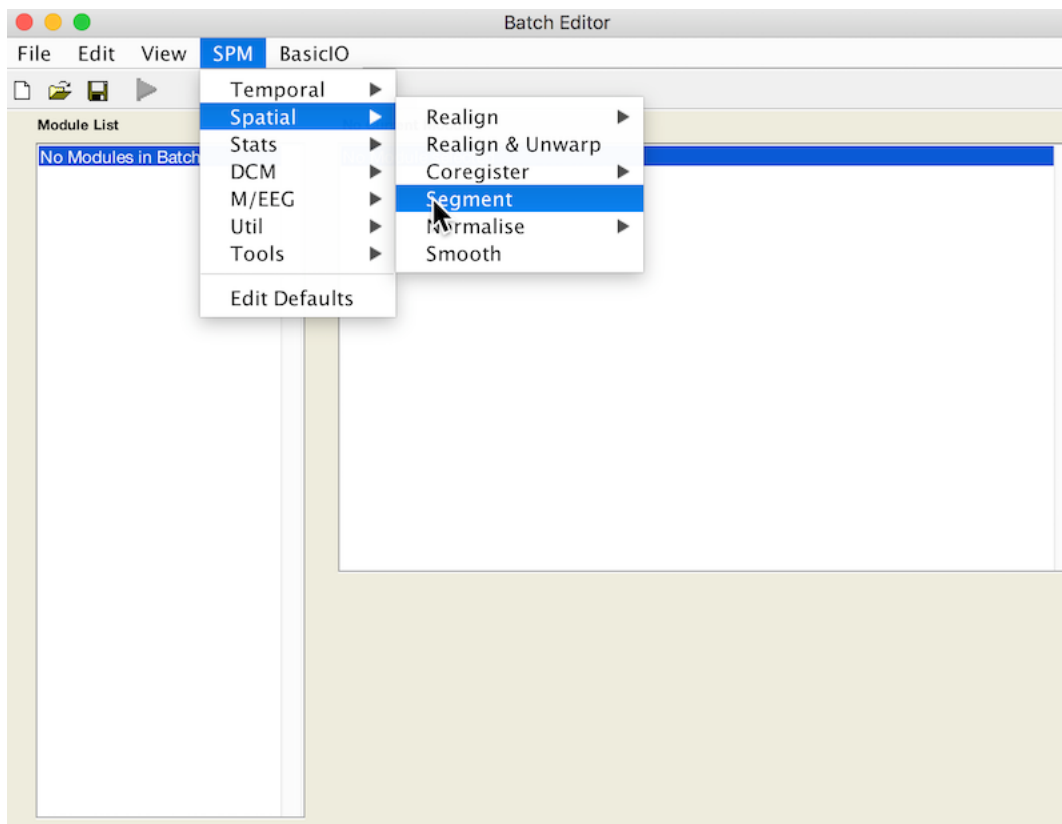
- Under *Volumes*, an **X** is displayed, indicating that this parameter must be specified. Specify the native space anatomical image (whole head) that you used to draw the lesion (remember to pick the simplified form, if you want to facilitate scripting).

- Scroll down to the bottom of the module options, and set *Deformation Fields* to *Forward*. The *Forward* warp will be used to warp files from native space into standard space. If you elect to create an Inverse deformation as well, choose *Inverse + Forward*. That would create an additional warp allowing you to deform files from standard space into native space.

- This is your first batch module. Now we'll add another.

Normalise: Apply Warp Field to Lesion Mask

1. From the *Batch* editor, choose: *SPM Spatial Normalise Normalise: Write*

2. This adds a second module to your batch

3. Double click *Data*. *Deformation Field* and *Image to Write* now appear in the menu.

4. For *Deformation Field*, click *Dependency* and select *Segment: Forward Deformation* to specify the **y** file, even though you have yet to create that file. Click OK.

- Under *Image to Write*, specify the native space lesion file that you want to warp.

- Under *Interpolation* choose *Nearest neighbour* This will insure the lesion mask keeps mask values at or near 0 and 1.

Normalise: Apply Warp Field to Anatomical Image

- You should apply the warp field to the original native space structural image. This is at minimum a sanity check because you can easily see inappropriate deformations of the whole head that you might not notice in the lesion file alone.

- Alternatively, if you have not created a lesion mask, then you can draw the lesion mask in standard space on the results of this section.

1. From the *Batch* editor, choose: *SPM Spatial Normalise Normalise: Write*

2. This adds a module to your batch

3. Double click *Data*. *Deformation Field* and *Image to Write* now appear in the menu. Add the native space image for *Image to Write*.

4. For *Deformation Field*, click *Dependency* and select *Segment: Forward Deformation* to specify the **y** file, even though you have yet to create that file. Click OK.

- Use the default interpolation: *4th degree B-Spline*.

- All other values should be left as their defaults.

Run Batch and View Results

- Run the batch module by clicking the green go button (upper left on batch editor).

- Segmentation produces:

    - Files prefixed with **c** for each tissue type and the skull.

- A *seg8.mat* file containing segmentation parameters.

  - A deformation field image with **y** prepended.

  - If you elected to create an Inverse deformation as well, that will be prefixed with **iy**.

- Normalization produces:

  - A lesion image in standard MNI space. It has a **w** prefixed to the name to indicate that it has been warped.

  - A structural image in standard MNI space, also with a **w** prepended.

8.3.4 SPM Lesion Normalization with Tissue Probability Maps

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2018_06_25
Date Updated: 2019_06_26
Tags: SPM, normalization, standard space, lesion, anat, TPM
Software: At least Matlab 2016B, SPM12

Goal: Warp a native space lesion mask (and optionally a native space anatomical image) into standard (MNI) space. In this procedure we use the lesion mask to define an additional tissue probability map (TPM) that influences the warp.

Data

- Images must **not** be zipped (i.e., *T1w.nii* is okay, but *T1w.nii.gz* is not).

- An anatomical (anatomical) image, usually T1 weighted, but could be T2 or flair weighted.

- A lesion mask with values of 1 in the lesion and 0 elsewhere.

- See *Batch and Scripting* for suggestions about file naming that will make it easier to script your resulting batch job.

Steps

Coregister: Reslice Lesion Mask to TPM Space

- The lesion mask must be in TPM space before it can be used to define the additional tissue type.

- Open Matlab (and optionally navigate to the relevant directory).

- Type *spm* at the Matlab command prompt. This will open spm

- Click the *fMRI* button which will open the windows you need.

- Select *Batch* and from the menubar: *SPM Spatial Coregister Coregister: Reslice*.
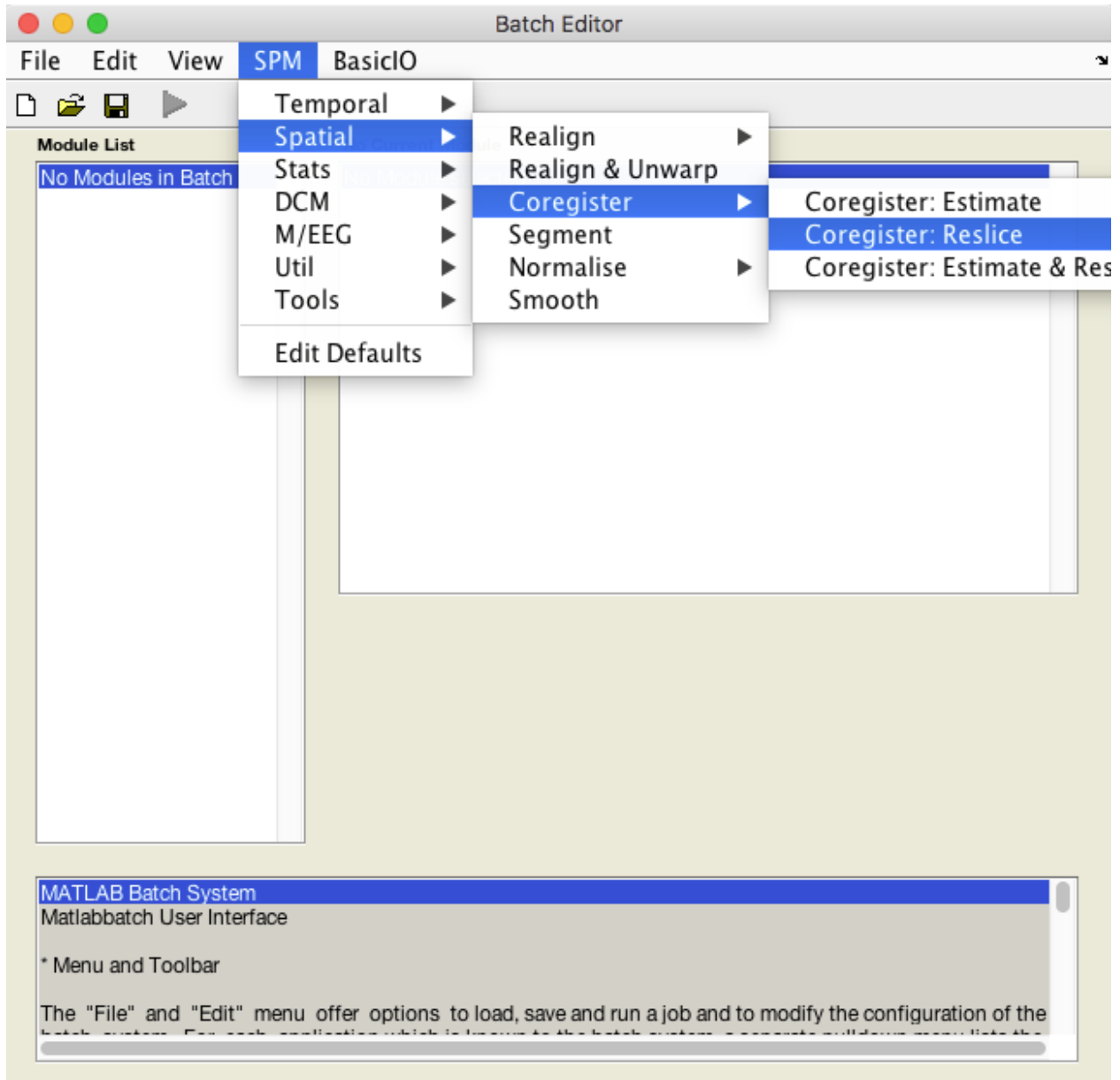
- An **X** is displayed for the two parameters that must be specified: *Image Defining Space* and *Images to Reslice*.

- The *Image Defining Space* is the TPM.nii, typically found in the *spm12/tpm* directory.

- The *Images to Reslice* will be the lesion mask.

- Under *Interpolation* choose *Nearest neighbour* This will insure the lesion mask keeps mask values at or near 0 and 1.

- All other values should be left as their defaults.

- This is the first module of your batch job.

Segment: Estimate Warp Field

- From the batch editor choose: *SPM Spatial Segment*

- Under *Volumes*, an **X** is displayed, indicating that this parameter must be specified. Specify the native space anatomical image that you used to draw the lesion.

- Click *Tissues* in the module options.

- Under *Current Items: Tissue*, click *New Tissue*.

- Scroll to the bottom of the tissue list, where you should find an **X** for *Tissue probability map*.

- Choose *Dependency* (red box). In the popup *Tissue Probability map* choose *Coregister: Reslice: Resliced Images* (blue box). This is how you refer to the resliced lesion map **r*** that you have not yet created.

- Scroll down to the bottom of the module options, and set *Deformation Fields* to *Forward*. The *Forward* warp will be used to warp native space images into standard space.

- All other values should be left as their defaults.
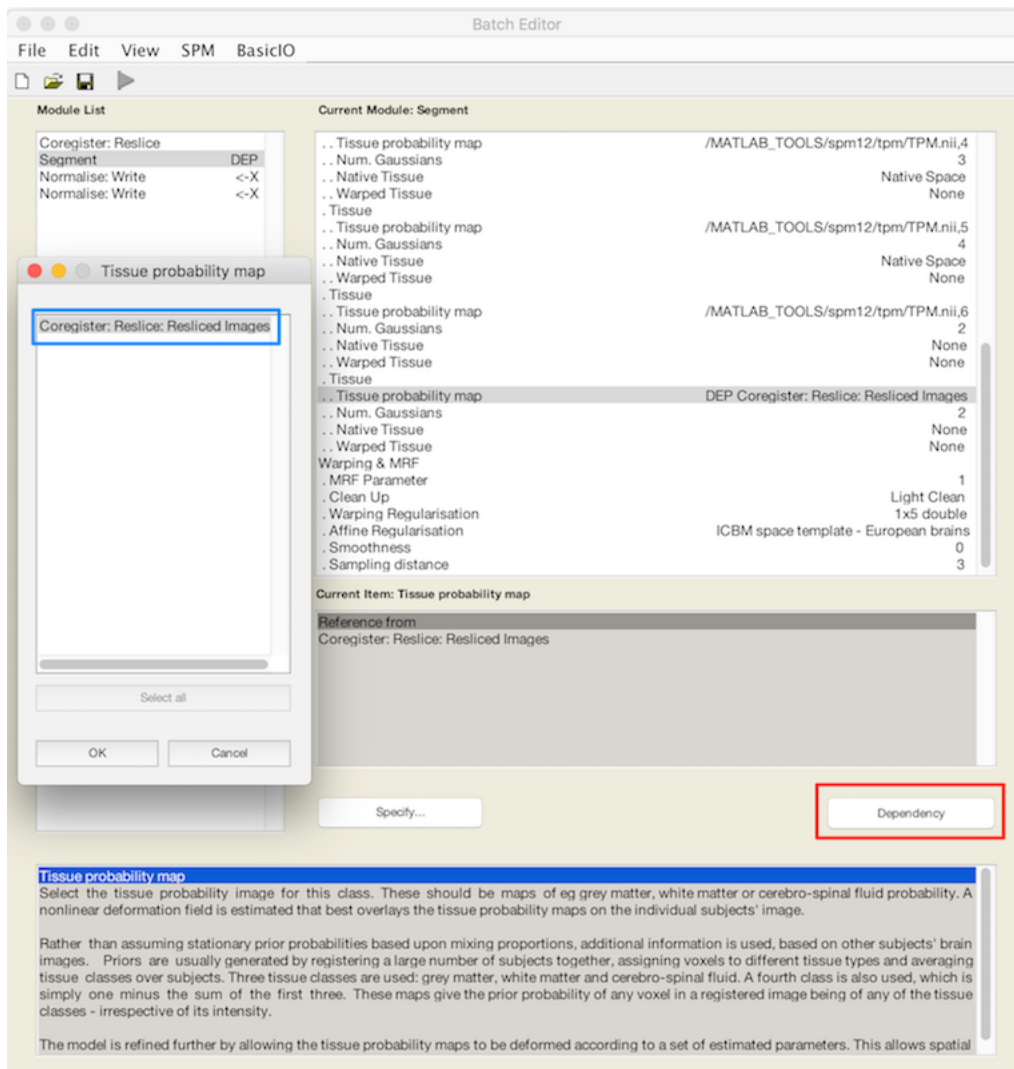
- This is the second module of your batch job.

Normalise: Apply the Warp Field to Lesion Mask

1. From the batch editor choose: *SPM Spatial Normalise Normalise: Write*

2. Double click *Data*. *Deformation Field* and *Image to Write* now appear in the menu.

3. For *Deformation Field*, choose the *Dependency* button and specify *Segment: Forward deformations*

4. Under *Image to Write*, specify the native space lesion file that you want to warp.

- Under *Interpolation* choose *Nearest neighbour* This will insure the lesion mask keeps mask values at or near 0 and 1.

- All other values should be left as their defaults.

- This is the third module of your batch job.

Normalise: Apply Warp Field to Brain and Head

- You should apply the warp field to the original native space anatomical image as well. This is a sanity check because you can easily see inappropriate deformations of the whole head that you might not notice in the lesion file alone.

- Repeat the first four steps above to add a fourth batch module.

- Add the native space image for *Image to Write*.

- Use the default interpolation: *4th degree B-Spline*.

- All other values should be left as their defaults.

- This is the fourth module of your batch job.

Run Batch and View Results

- Run the batch module by clicking the green go button (upper left on batch editor).

- Reslicing produces a new lesion file prefixed with **r** to indicate it has been resliced.

- Segmentation produces:

  - Files prefixed with **c** for each tissue type and the skull.

  - A *seg8.mat* file containing segmentation parameters.

  - A deformation field image with **y** prepended.

  - If you elected to create an Inverse deformation as well, that will be prefixed with **iy**.

- Normalization produces:

  - A lesion image in standard MNI space. It has a **w** prefixed to the name to indicate that it has been warped.

  - An anatomical image in standard MNI space, also with a **w** prepended.

8.3.5 SPM Lesion Normalization with Source Weighting Image

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2018_07_10
Date Updated: 2019_06_26
Tags: SPM, normalization, standard space, lesion, anat, TPM
Software: At least Matlab 2016B, SPM12

Goal: Warp a native space lesion mask (and optionally a native space anatomical image) into standard (MNI) space. In this procedure we use an inverse lesion mask to define areas that should influence the warp. The old normalize procedure did not depend on an initial segmentation and thus had distinct disadvantages over the newer registration that matches tissue type boundaries.

Data

- Images must **not** be zipped (i.e., *T1w.nii* is okay, but *T1w.nii.gz* is not).

- An anatomical (anatomical) image, usually T1 weighted, but could be T2 or flair weighted.

- A lesion mask with values of 1 in the lesion and 0 elsewhere.

Steps

Create Inverse Lesion Mask

If you have a mask with 1 anywhere there is lesion and 0 elsewhere, you need the inverse of that. To create this inverse mask in SPM12, you can use the image calculator (N.B. Although the resulting mask displayed in FSL shows the "1" to be slightly more or less than 1, SPM nevertheless seems to treat the result as a true binary mask). You have two choices for accessing the image calculator:

- Choose ImCalc from the main SPM menu
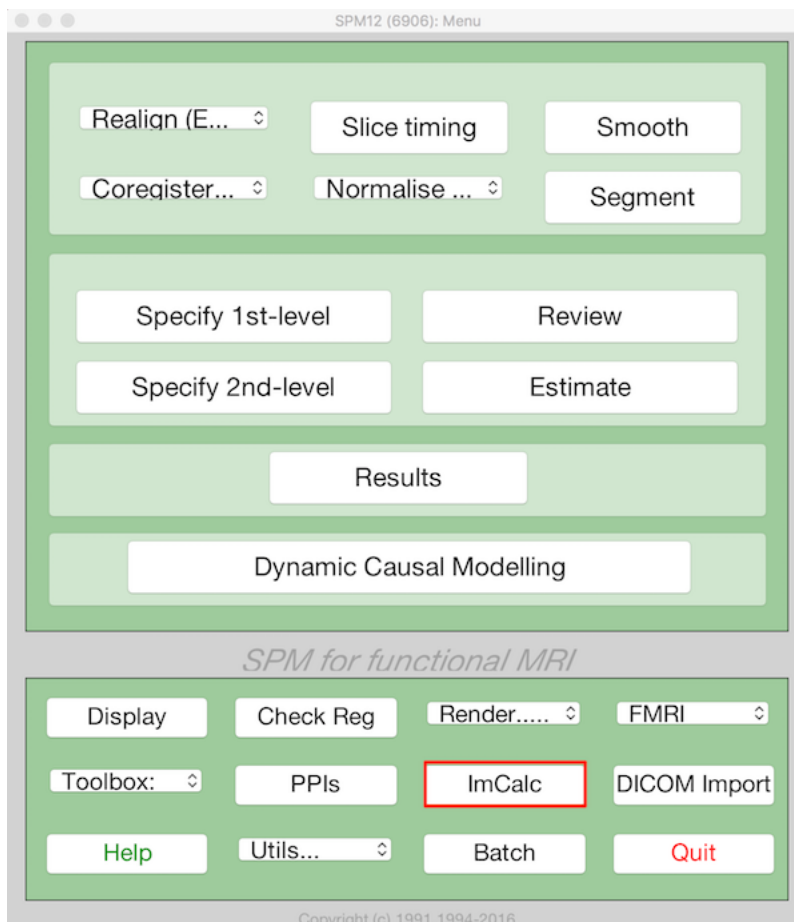
- Select *Batch* and from the menubar: *SPM Util Image Calculator*
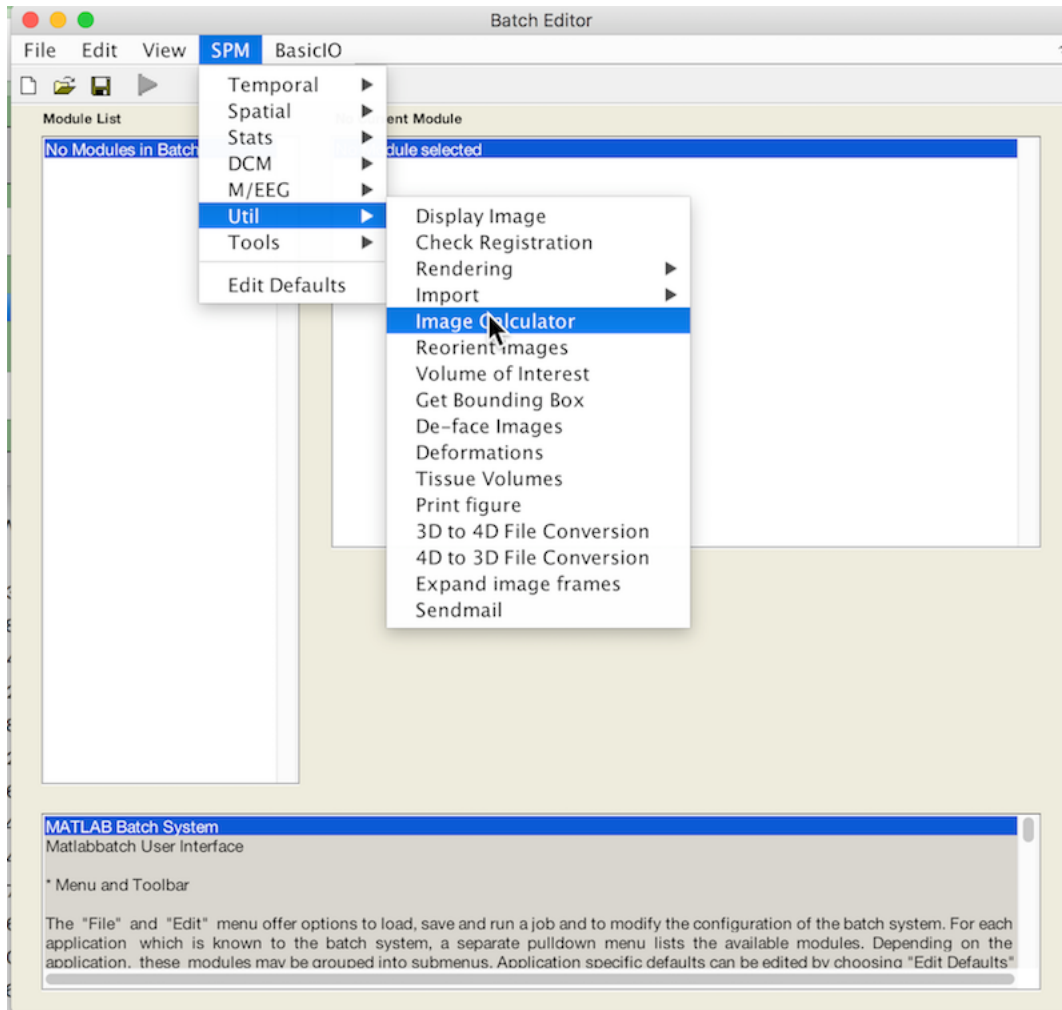
- An **X** is displayed for the two parameters that must be specified: *Input Images* and *Expression*. You may also change the Output filename and directory.

- The *Input Image* will be the native space lesion mask

- The expression will be **(i1 .* -1)+1**

- Set *Interpolation* to Nearest Neighbor to prevent values between 0 and 1 from being created at the edges of the lesion.

- Set *Data Type* to *UINT8 -unsigned char*
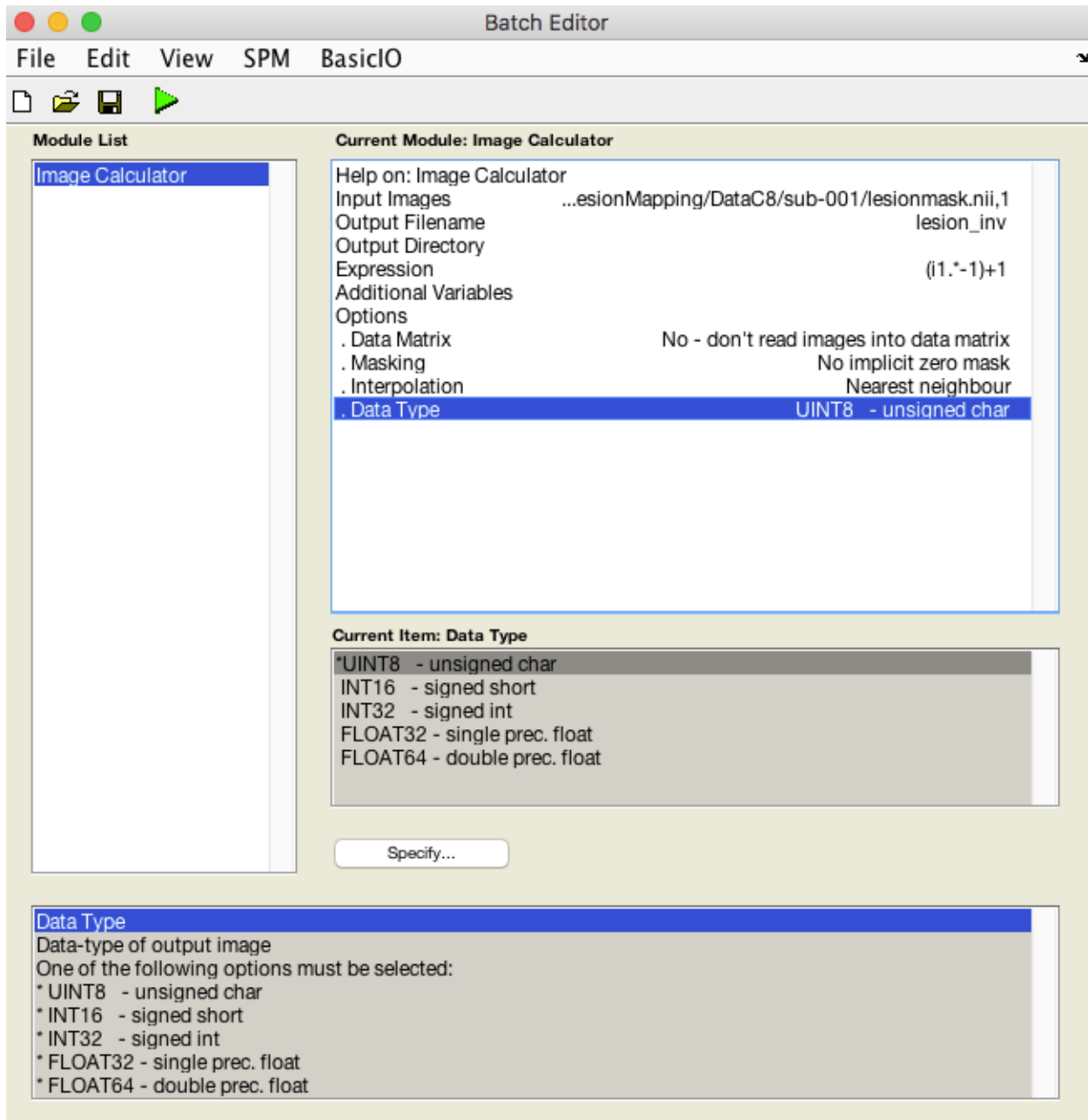
- In the image calculator, *i1* always refers to the first input image you selected (i2= the second image etc.). In this expression we multiply all values in *i1* by -1, so 0 remains 0, but 1 becomes -1. This inverts the values.Then we add 1 to all voxels the resulting image so that the range is 0 to 1 and the values have been inverted.

- This is the first module of your batch.

Old Normalize: Estimate and Write Lesion Mask

- Select *Batch* and from the menubar: *SPM Tools Old Normalise Old Normalise: Estimate and Write*

- Click *Data*, and select *New Subject* from the grey *Current Item: Data* box below. **X** values are displayed, indicating parameters that must be specified.

- Specify the native space anatomical image that you used to draw the lesion as the *Source Image*. * We also want a *Source Weighting Image*. Click *Source Weighting Image* and select the *Dependency* button. Choose the *Image Calculator: ImCalc Computed Image* for your inverse lesion file.

- For *Images to Write* select your original lesion file.

- Under *Estimation Options* Select a *Template Image* from the OldNorm subdirectory, choose a template with the same contrast as your source image (i.e., if our source image is a T1, choose T1.nii.)

- Under *Writing Options* find *Interpolation* and select *Nearest neighbour*
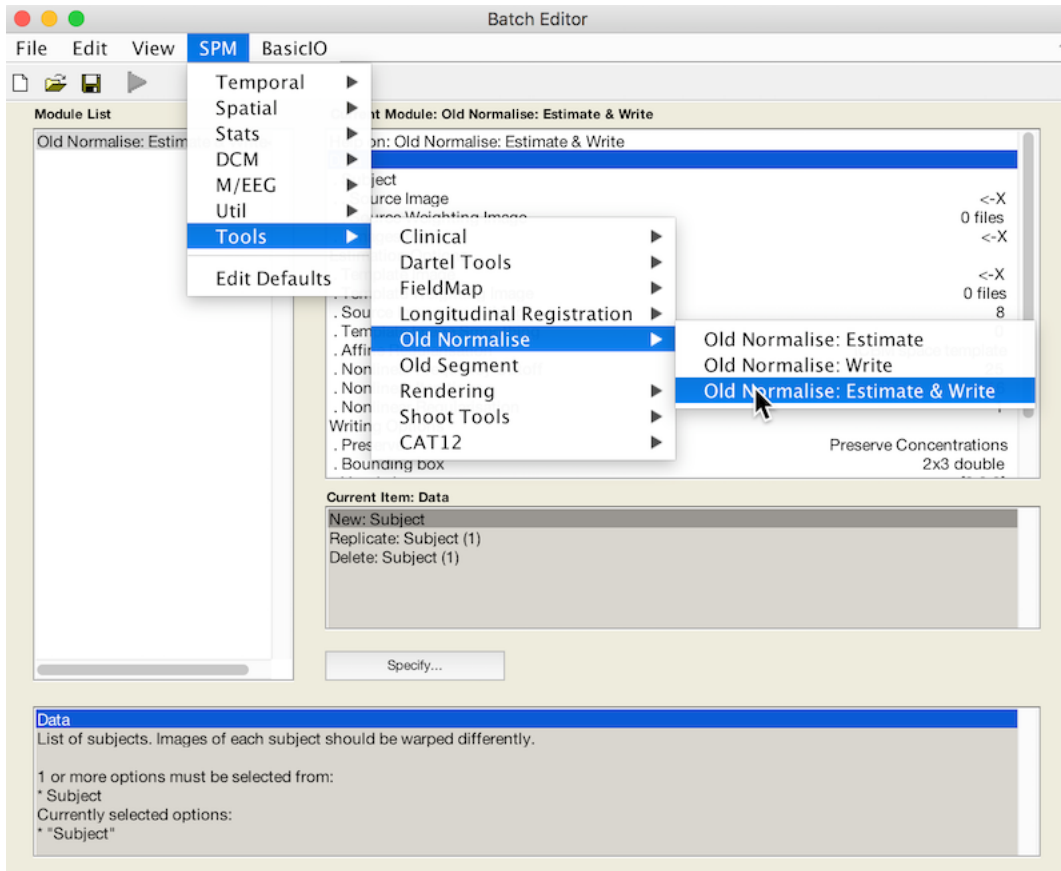
- This is the second module of your batch.

Old Normalize: Write Anatomy file

- From the batch editor, choose: *SPM Tools Old Normalise Old Normalise: Write*

- For the *Parameter file* select *Dependency* and the only *Norm Params File* available.

- Select the anatomical file for *Images to Write*

- Leave the default trilinear interpolation for this one.

- All other values should be left as their defaults.

Run Batch and View Results

- Run the batch module by clicking the green go button (upper left on batch editor).

- The ImCalc module will generate the inverse lesion mask.

- Normalization produces warped files prefixed with *w*.

- In addition, a *_sn.mat* file is produced that can be used to write additional warped images.

A test of several normalization methods was completed with 27 subjects selected from the publicly available dataset described here:

Liew, S.-L., Anglin, J. M., Banks, N. W., Sondag, M., Ito, K. L., Kim, H., et al. (2018). A large, open source dataset of stroke anatomical brain images and manual lesion segmentations. Scientific Data, 5, 180011–11. http://doi.org/10.1038/sdata.2018.11

A pdf presentation comparing these methods is available here: 10_things_lesions.pdf or 10 things lesions.pptx: The dataset was sufficiently large, homogenous, and well-documented to provide a grounding for comparing techniques.

8.3.6 Exploratory Comparisons as Tableau Dashboards

Tableau dashboards are interactive and dynamic. In the two dashboards described below, you can change which subjects and which methods you are viewing with the filters to the right of the dashboard. Mousing over features (i.e., bars in bar charts, cells in tables, lines and circles in the scatterplot) displays more information.

Brain Mask Matching Dashboard

This first dashboard explores how each method succeeds in matching the patient's brain mask to the standard brain mask. Because we want to compare patients and determine whether their lesions are in the same place, it is important for their masks to match the standard (and therefore, each other). Obviously this is a very rough measure of how completely the brains are matched in standard space. Note that **FSL** does especially well on this measure: high Jaccard overlap values in the hotspot table, high mean value in the top right bar chart and low standard deviation in the bottom right bar chart: Brain Mask Matching

Lesion Preservation Dashboard

The second dashboard explores how well lesion volume is maintained after normalization. For example, if the lesion occupied 15% of the brain mask in the native space image and 18% of the brain mask in standard space, then we'd say there was a 3% increase (pink cell in table) in the size of the lesion as the result of the normalization process. What we want is for lesion volume to stay the same (grey cells). Scroll down in the dashboard table to see the few instances that exceed 1% (sub-027 and sub-030).

What we fear is that larger lesions are more affected by the normalization process than smaller lesions. At the bottom of the dashboard is a scatterplot with best fit linear model lines for each method. Indeed, there is a significant effect of lesion volume on the normalization process for every method except **ant2**. However, the change in lesion volume that results from normalization is VERY small. This can be seen by examining the standard deviation and mean of the changes (shown at the top of the dashboard): mean < 1% ; std < 2% for all the methods. Lesion Volume Preservation

8.3.7 Overview of Results

In general, modern methods work well (ant2, ant3, fsl, spm_head, spm_tpm_head) though several important points deserve notice:

- SPM12 does MUCH better with the whole head than with the extracted brain.

- There isn't any great advantage to providing SPM12 with a lesion mask.

- The SPM12 approach to normalization is very different than ANTS or FSL.

- ANTS and FSL appreciate a GOOD brain extraction.

- Brain extraction is tricky for brains with big lesions. See the optiBET discussion on the *FSL Lesion Normalization* page.

8.3.8 Recommended Approach

Generally, you can follow this description *SPM12 Normalization*. However, if you have not yet created a lesion mask, then you can implement the steps described in the *Segmentation section* and the *Normalise Anatomical Image section*, but skip the step where you normalize the lesion mask (because, after all, in this scenario you do not have a lesion mask).

If you are interested in working with a tool like tractotron, then you'll need to crop your result files into FSL standard space. This is easily accomplished with the spm2fsl.sh script. A description is available here *Convert Images from SPM MNI to FSL MNI*.

In addition, you may find that SPM introduces NaNs into your image. NaNs can cause trouble in many programs, so *Ensure Zeros, not NaNs*.

8.3.9 Normalization Methods Compared

ANTS

- See *ANTS Lesion Normalization*

- **ant1** ants version 1.X normalization routine. Start data is the result of optibet.sh (see FSL section below), hence skull-stripped. No source weighting. Normalized to MNI152_T1_2mm_brain.nii.gz.

- **ant2** ants version 2.X normalization routine. Start data is the result of optibet.sh, hence skull-stripped. Source weighting. Normalized to MNI152_T1_2mm_brain.nii.gz.

- **ant3** Same as ant2, except start data was the final result of fsl_anat_alt, rather than the result of optibet. Lesion preservation comparisons are based on the final fsl_anat_alt mask.

FSL

- See *FSL Lesion Normalization*

- **fsl** Runs modified fsl_anat to turn on the lesionmask option, and to run optibet early on to get a much better initial skull-strip. This allowed us to use the skull-stripped image for segmentation and normalization. Tests showed optibet did a much better job of skull-stripping than bet. Source weighting. Normalized to MNI152_T1_2mm_brain.nii.gz. Lesion preservation comparisons are based on the final fsl_anat_alt mask.

SPM

SPM: Old

- See *SPM Old Normalization*

- **spm_old** Old normalization. Normalized to Template Image/T1.nii. source weighting, whole head.

- **spm_old_nosw** Old normalization. Normalized to Template Image/T1.nii. No source weighting. whole head.

SPM12

- See *SPM12 Normalization*
- **spm_brain** spm12 normalization starting with the optibet result (hence skull-stripped). No source weighting.
- **spm_head** spm12 normalization starting with the whole head. No source weighting.

SPM12 TPM

- See *SPM12 Normalization with TPM*
- **spm_tpm_brain** spm12 normalization starting with the optibet result (hence skull-stripped). Defines lesion as tissue type.
- **spm_tpm_head** normalization starting with whole head. Defines lesion as tissue type.

TMS

9.1 Introduction

This page describes the TMS system at the University of Arizona. This equipment belongs to the Chou lab. To request access, contact Ying-hui Chou (yinghuichou at email.arizona.edu) to discuss your proposal. The equipment resides in a locked room in the basement of the BSRL, down the hall from the MRI scanner.

- If you find inaccuracies (or typos or broken links), please let me know: dkp at email.arizona.edu. The descriptions provided below are based on extensive and excellent lab manuals prepared by Yu-chin Chen and Aneta Kielar. This page should provide you with a good sense of how you might proceed in a typical study, but it is not meant to be exhaustive. There is, for example, a 200+ page manual on the TMS-Navigator software (sorry, this is proprietary and cannot be posted).

- It might be helpful to have some kind of TMS Checklist. The linked one is in Word format so you can edit it to add or remove the details you need.

A short (3.5 minute) TMS Experience movie is available to show potential participants what it is like to get set up for TMS.

Want PDFS? Read the Docs has the ability to output documentation in other formats: pdf and epub. On the bottom left it says `Read the Docs v:latest`. Under `Downloads`, choose one of the `latest` options (e.g. PDF). The PDFs have much nicer formatting than if you simply "save as" pdf in your browser. The only downside is that the pdf includes everything on the website, so you'll want to use Preview or Adobe Acrobat or something similar to extract the portion of the documentation you want. Of course, you could just print the range of pages you want as hard copies.

9.2 Hardware

To use the TMS (Transcranial Magnetic Stimulation) for research, we rely on three separate hardware systems. Here they are:

- MagPro
- Localite Computer and Camera
- MEP (Motor Evoked Potential)

9.2.1 MagPro
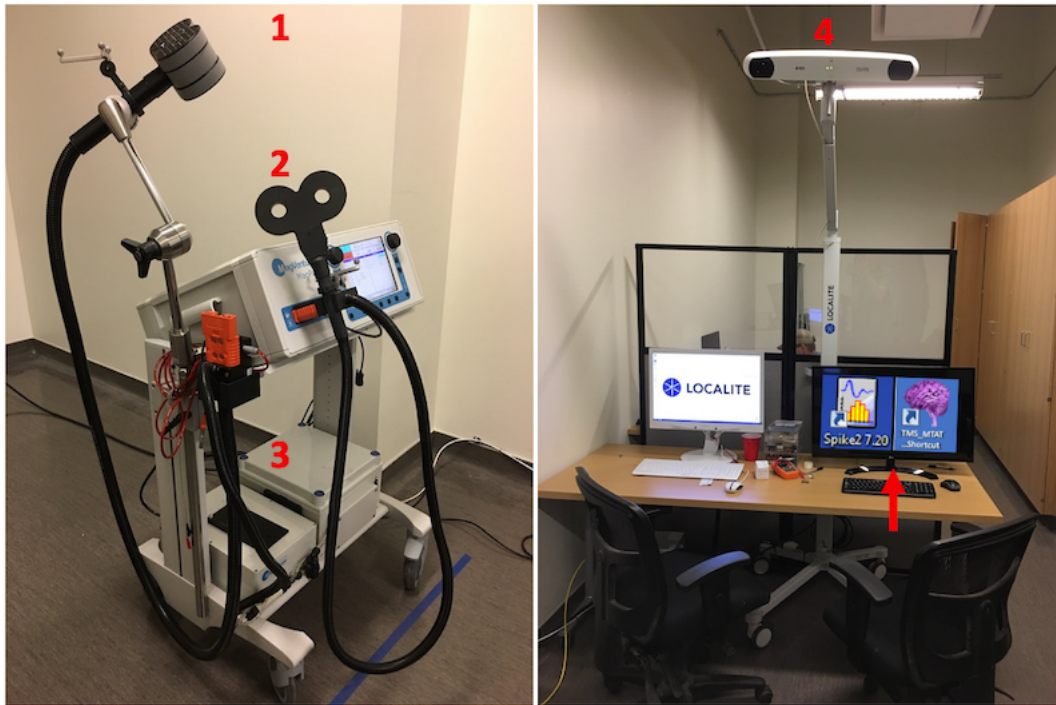
Fig. 1: **Left**: The MagPro implements desired pulse sequences. The small C-B60 coil 2 administers single pulses: useful for demonstration and for identifying the resting motor threshold. The large Cool-B65 coil 1 administers pulse trains. Administering pulse trains requires the liquid cooling unit 3 and associated isolation transformer on the bottom of the MagPro cart. **Right**: The Localite computer runs the TMS Navigator software and the associated Polaris Spectra P7 camera 4 tracks objects using infrared and specialized reflective reference balls. The MEP computer provides Spike 2 and Pest software. The MEP monitor power is controlled by a joystick on the back middle bottom of the monitor (**arrow**).

Left: The C-B60 coil being attached to the MagPro; **Right** the same coil being detached from the MagPro.
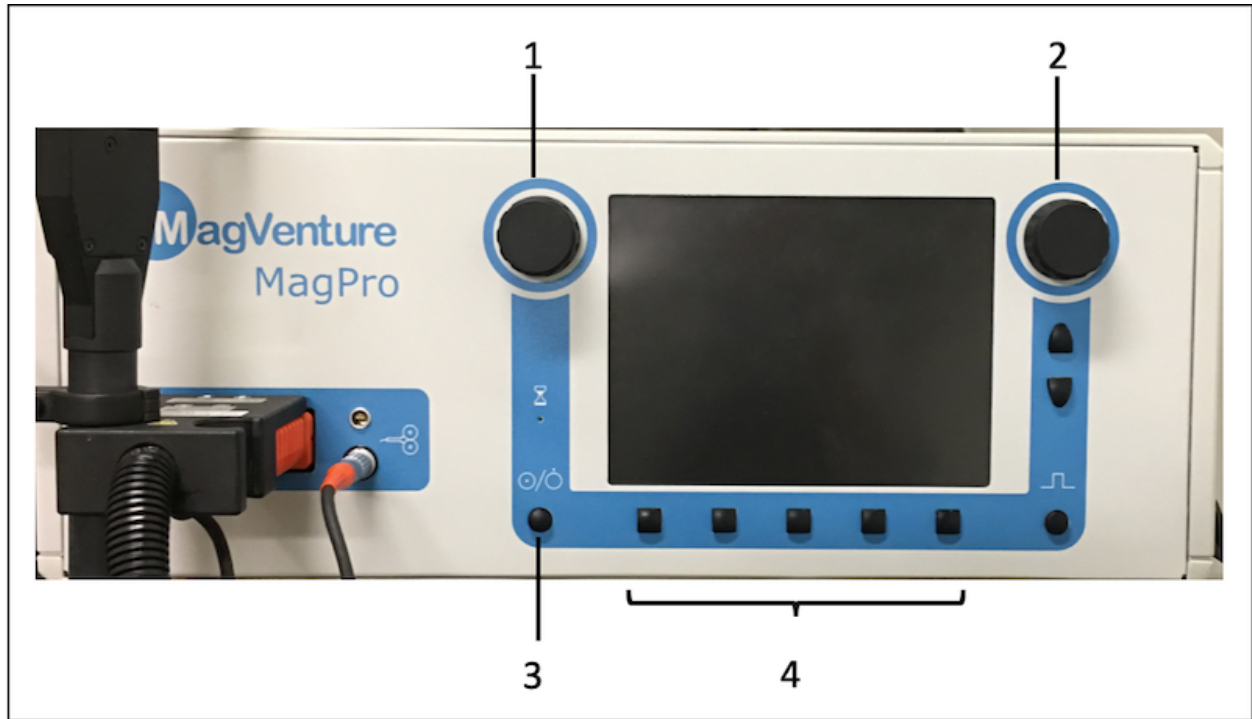
Fig. 2: MagPro Interface: 1) Amplitude Wheel; 2) Options Wheel; 3) Enable/Disable Button; 4) Soft Keys. The function of each soft key is shown in the display just above the key. The coil plugs in on the left.

9.2.2 MEP

9.3 TMS-Navigator Interface

2. **Display**: The view above (in the TMS Navigator Interface figure) displays four equal sized panels. To change the layout, choose *View Layout* in the menu. For example, you can maximize the size of panel 4 to make it easier to see the 3D representation. If the layout options are not available (e.g., while navigating) you can still zoom in the 3D panel (hold down the mouse wheel while moving the mouse).

4. The menu provides access to the Talairach definitions, MNI Planning, and lots of other things.

Fig. 3: MagPro Screen: 1=status area; 2=information area; 3=selection area; 4=soft key area. This figure displays
the default **Main** screen. 1: The amplitude wheel is one way to control the amplitude of stimulation (cyan field on
the upper left). The Enable/Disable button controls the **Status** of the coil: Disabled→Enabled. The status area
consists of fixed state fields, that do not change when you switch between **Main** and **Timing**. 3: The pulse sequence
name is displayed in the **Setup** field (cyan). The Options Wheel scrolls through the available pulse sequences. After
finding the desired pulse sequence, press the **Recall** soft key. 4: The leftmost soft key switches to the timing menu if
you wish to see details of the pulse sequence. Once in the timing menu, the leftmost soft key switches to say **Main**
so you can switch back to the Main display when finished. You can see an example of the Timing display in the *TBS
section*
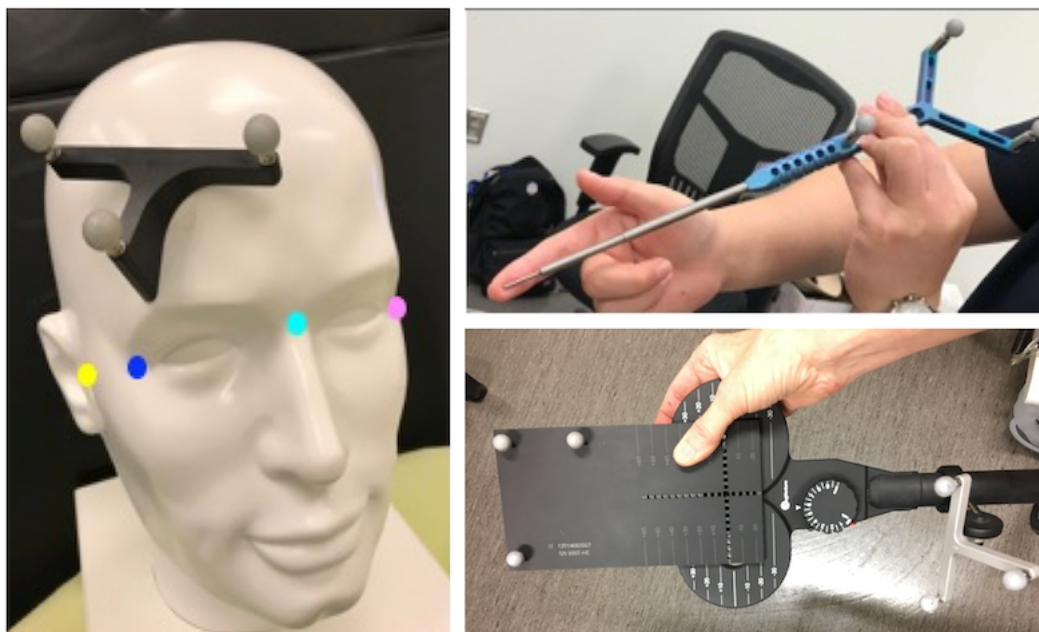
Fig. 4: Sets of three silver reference balls are attached to the participant's head **left**, the pointer **upper right**; the C-B60 coil and the calibration plate **lower right**. The three ball tracker on the participant's head is called the reference tracker. Try not to touch or nick the reference balls to keep them clean and undamaged. The pointer is very sharp: guide the pointer with your finger, covering the tip to prevent slipping/unwanted contact.
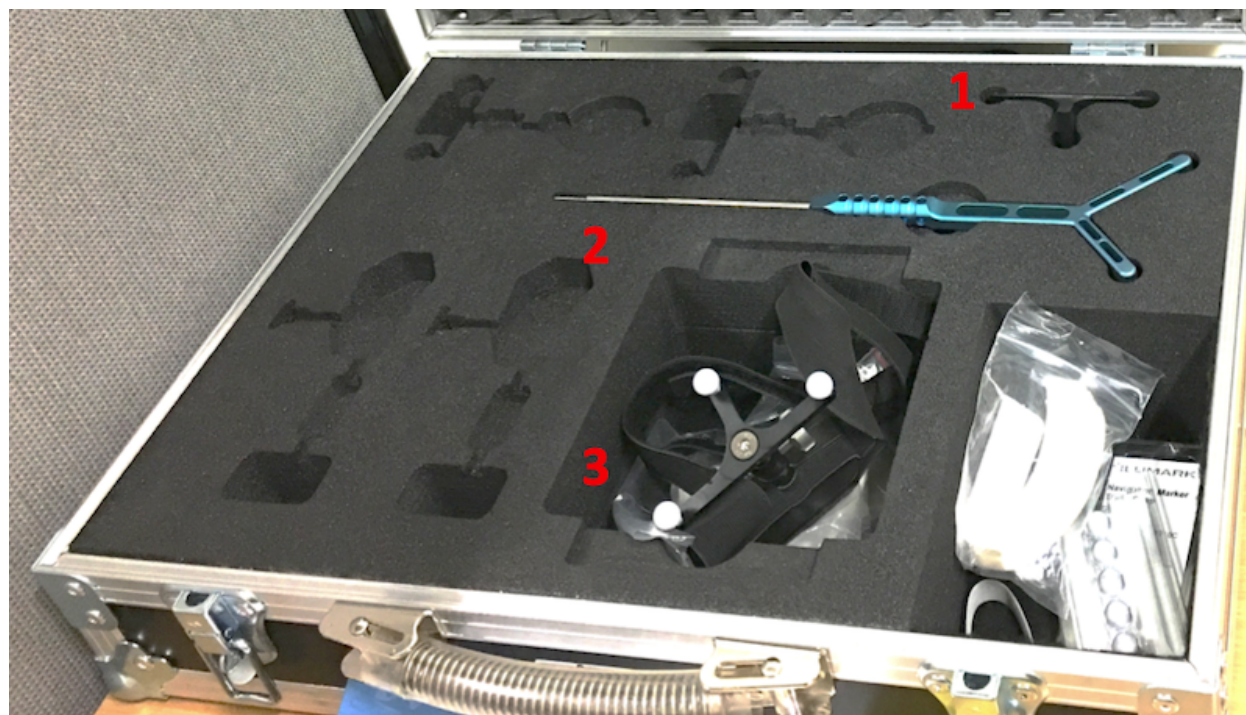
Fig. 5: The reflector balls are stored in a clean safe padded case: The stick-on reference tracker 1; the pointer 2; and the headband reference tracker 3.
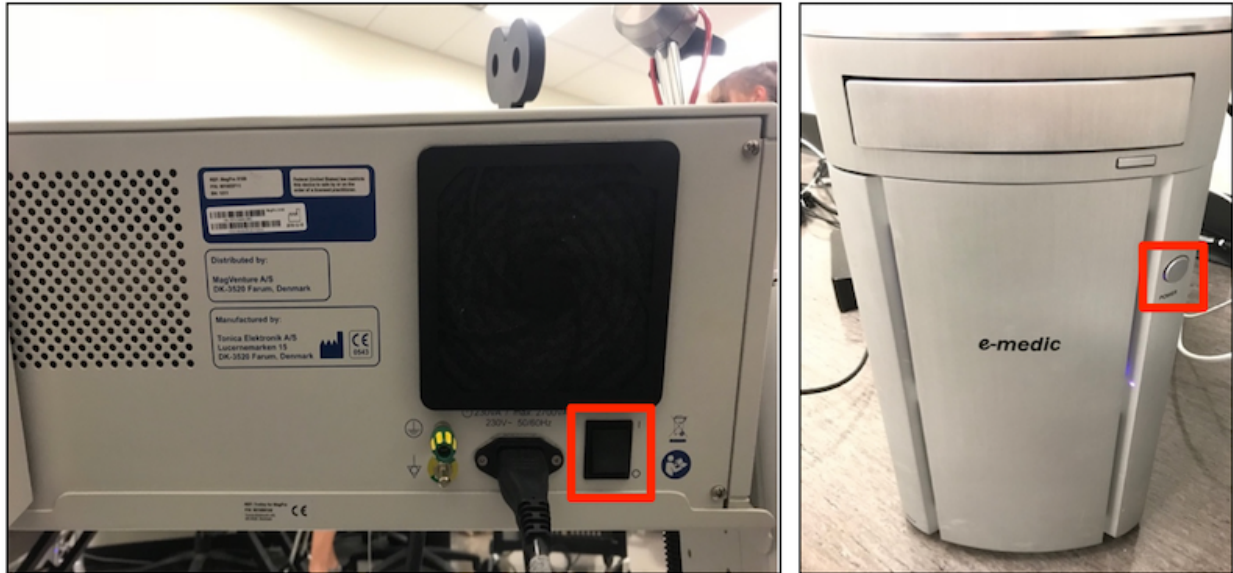
Fig. 6: **Left**: Turn on the MagPro (power switch is on the back, red rectangle) and allow it to boot up completely **before** turning on (**Right**) the Localite computer (under the table; power button in red rectangle) to which it is connected. This sequence ensures the Localite computer can detect the MagPro. In addition, before using the large Cool-B65 coil, turn on the cooling unit (else the coil will remain disabled). The cooling unit on the bottom of the MagPro cart, and its power toggle is on the front.

9.4 Once per Project: Folders and MNI Planning

Note: To create and edit MNI planning scenarios, a session does not have to be started or loaded, and the tracking system and instruments are not required.

- Create a folder on each computer (Localite and MEP) to store participant data for your lab.

- On the Localite computer, use **MNI Planning** to create one or more lists of stimulation targets in standard space:

    - Menu: File MNI Planning +

- Subsequently you can load these saved targets (You can always edit this later if necessary).

9.5 Flowcharts

Now you are somewhat familiar with the *TMS hardware* and *the TMS Navigator software*. Hopefully, you have set up *MNI Planning*. Now you can proceed to actually set up and run a participant. In the flowcharts below, clicking a node with a blue border will take you to the relevant section. Hardware=pink. Physical activities=cyan. Localite software=grey. MEP software=green. If order does not matter, a dotted line is used as a connector.
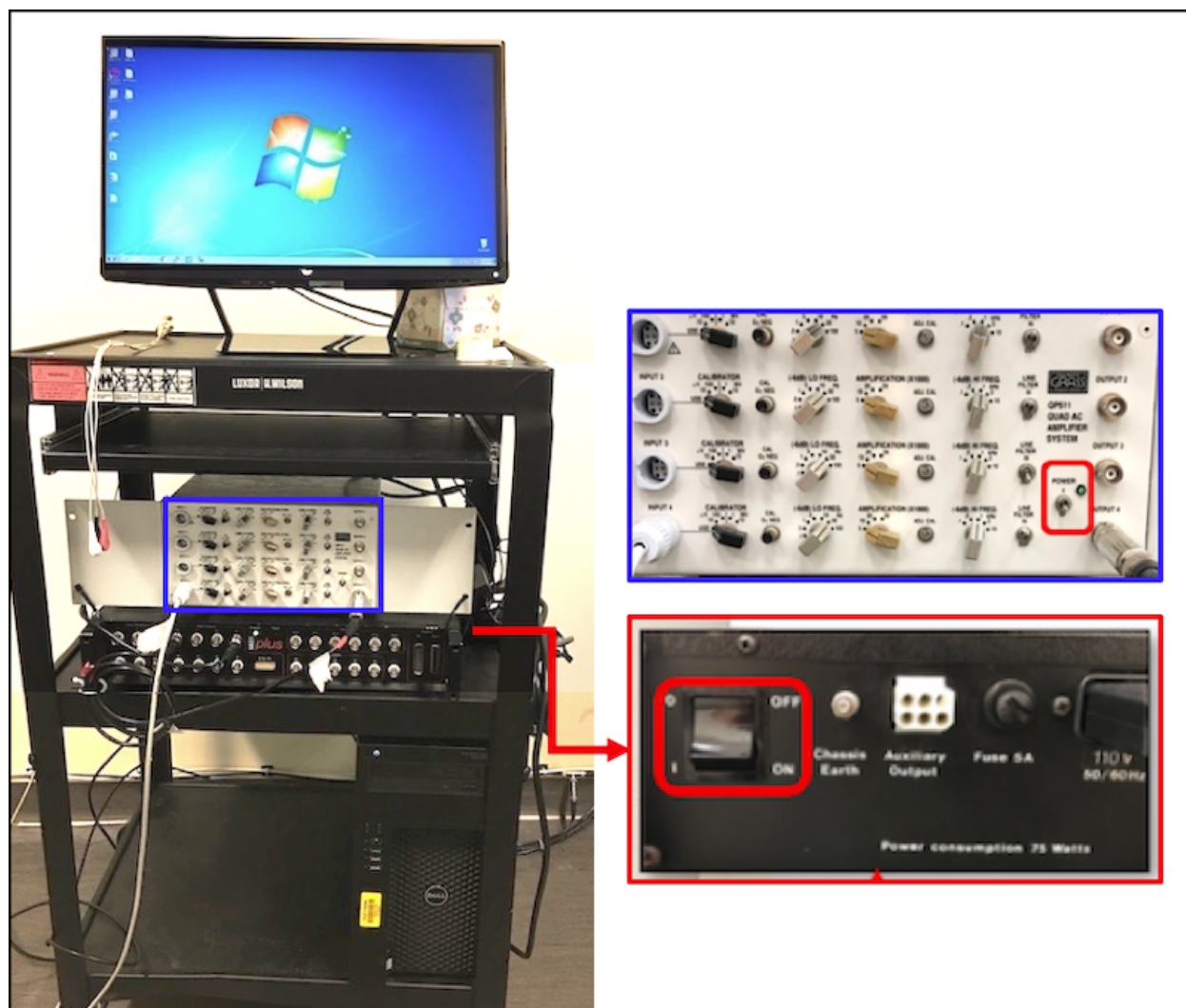
Fig. 7: **Left**: From top to bottom: The MEP (Muscle Evoked Potentials) system consists of an amplifier (white), data acquisition unit (black), and computer (black Dell) on a cart. **Right**: Power on white amplifier (switch on lower right) and black data acquisition unit (toggle on back of device). Generally, the MEP computer is already on, but power it on if it isn't. The monitor, keyboard and mouse for the MEP are on the table to the right of the Localite monitor.
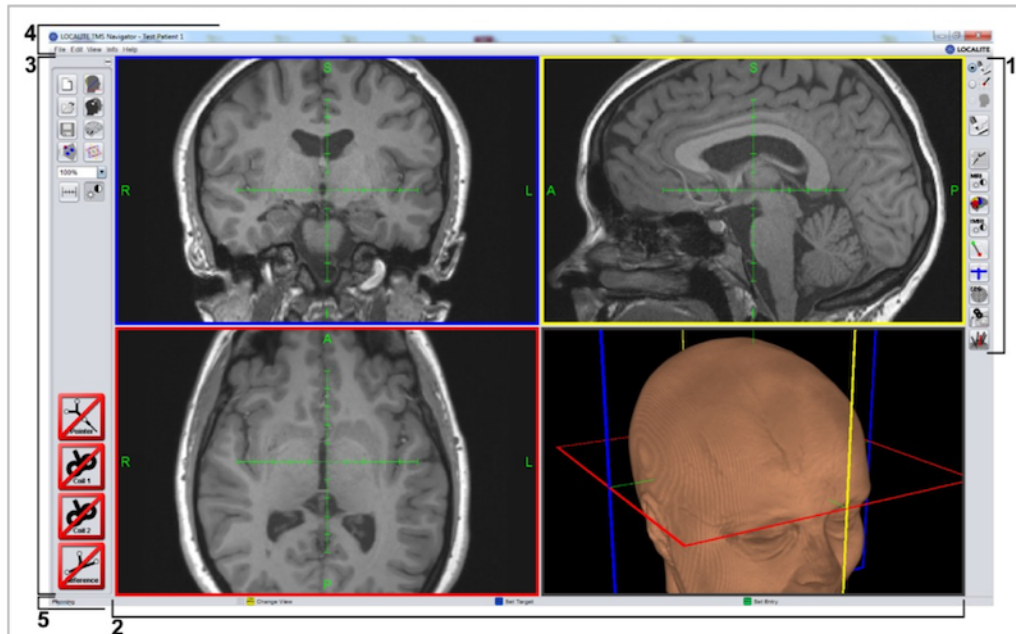
Fig. 8: Five important areas: 1. Control area; 2. Display; 3. Toolbar; 4. Menu; 5. Status bar

9.5.1 Before Participant Arrives

9.5.2 After Participant Arrives

9.6 Spike2 Setup (MEP computer)

Click the Spike2 icon to start the program. Spike2 is used to track muscle contractions from the electrodes you will place on the participant's hand.
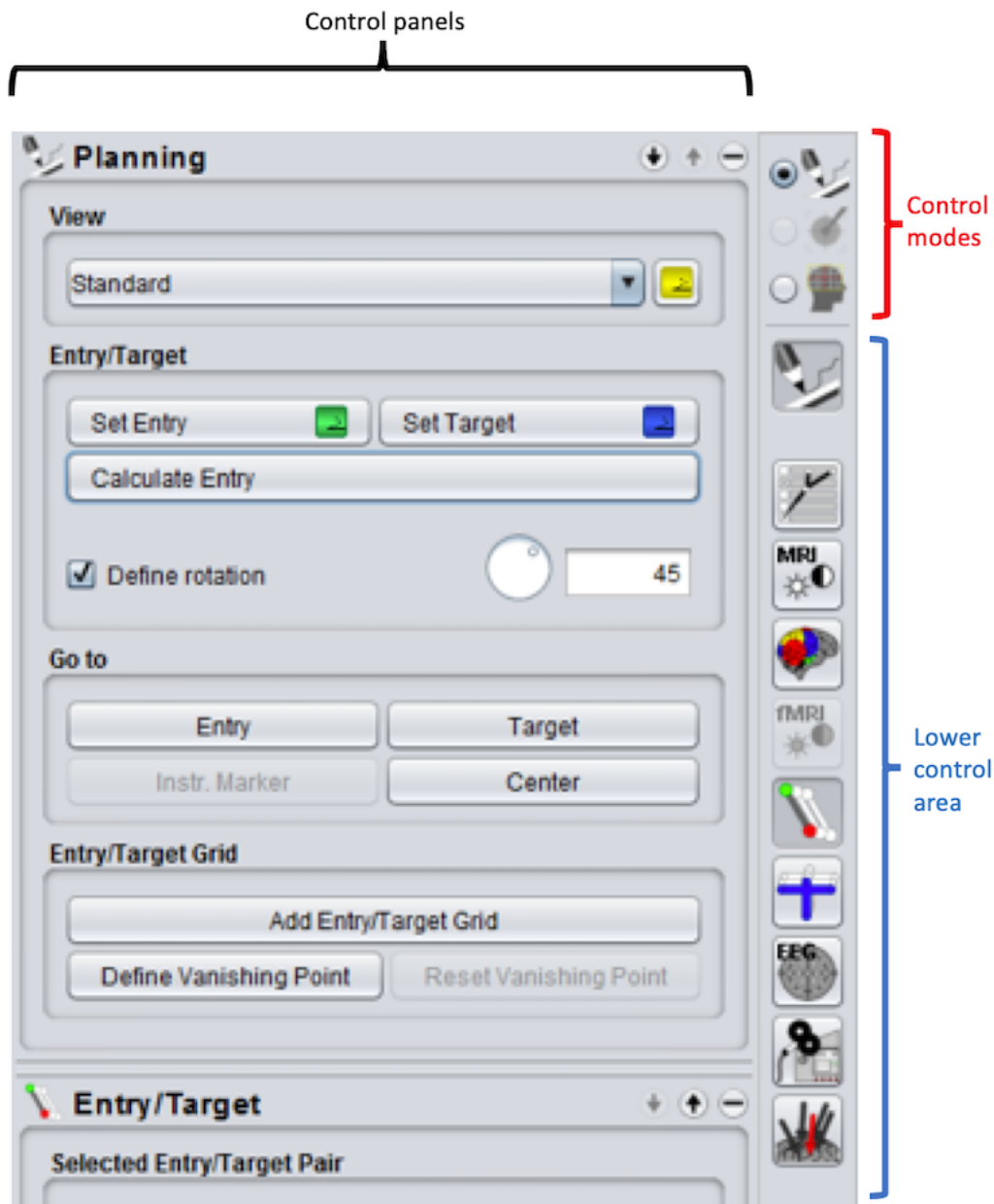
Fig. 9: 1. The **Control Area** is on the right of the display and divided into an upper **Control Modes** area and a **Lower Control Area**. Start with default **Planning Mode** (selected here). We will switch to **Navigation Mode** (by selecting the associated radio button) later. Clicking an icon in the **Lower Control Area** opens the corresponding **control panel** to the left of the Control Area. In this figure, we see the Planning control panel, and below that the title of the Entry/Target control panel. To the right of each control panel title are two arrows ↑ and ↓ to move the control panel up or down and a − to close the panel.
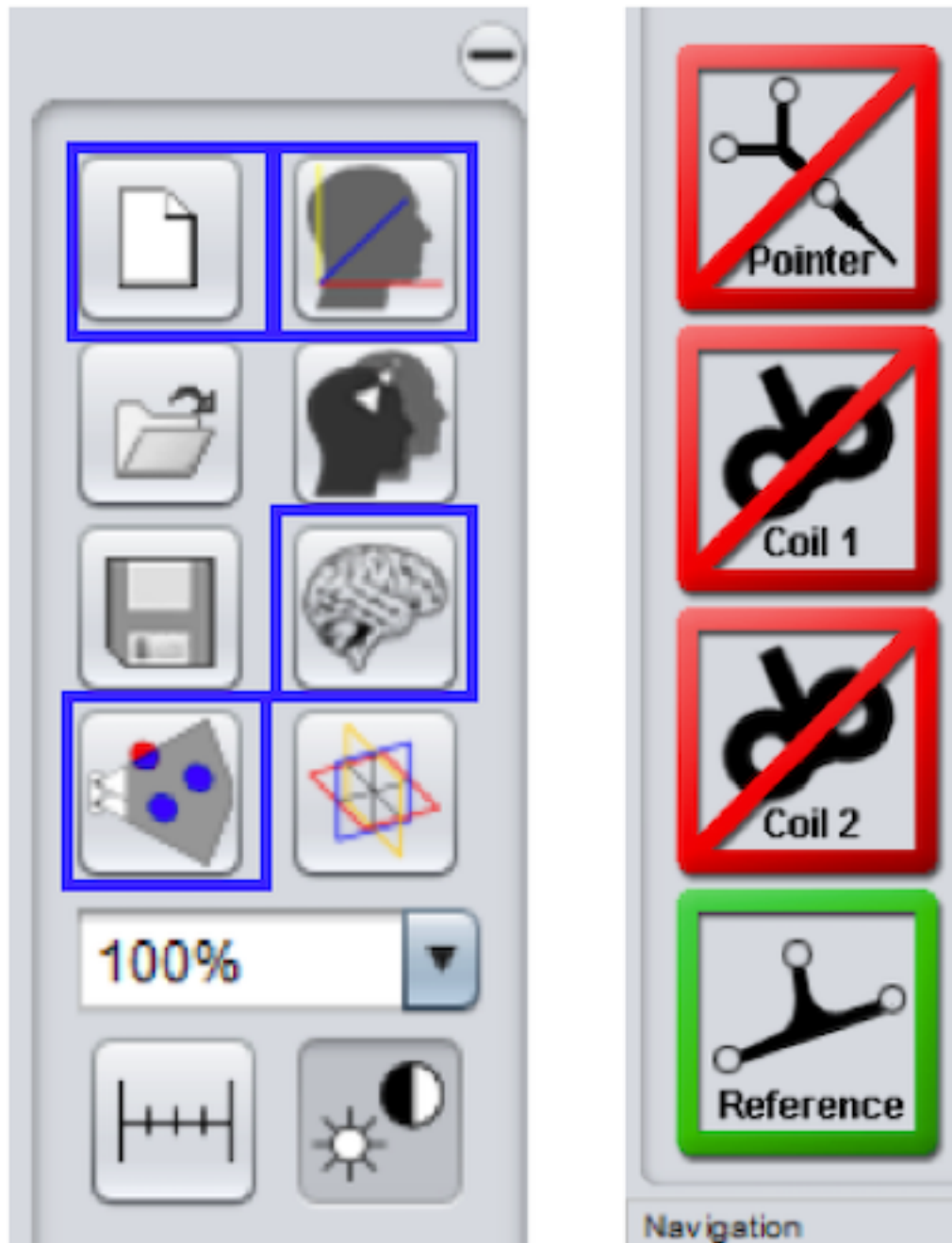
Fig. 10: 3. The **Toolbar** is on the left of the display area and divided into two sections illustrated above. **Left**: The upper section of the toolbar contains frequently used functions. We will discuss four of these functions in more detail (blue rectangles): setup session, patient registration, brain segmentation and camera check. **Right**: The lower section of the toolbar contains four icons that indicate whether devices are being detected by the camera (Red=No; Green=Yes): Pointer; Coil1; Coil2; Reference (on the participant's forehead).

Fig. 11: 5. The **Status bar** along the bottom of the interface displays tiny yellow, blue and green icons that describe the footpedal functions for the active mode. These same yellow, blue and green icons appear in various control panels where they implement the same functions as the footpedals.
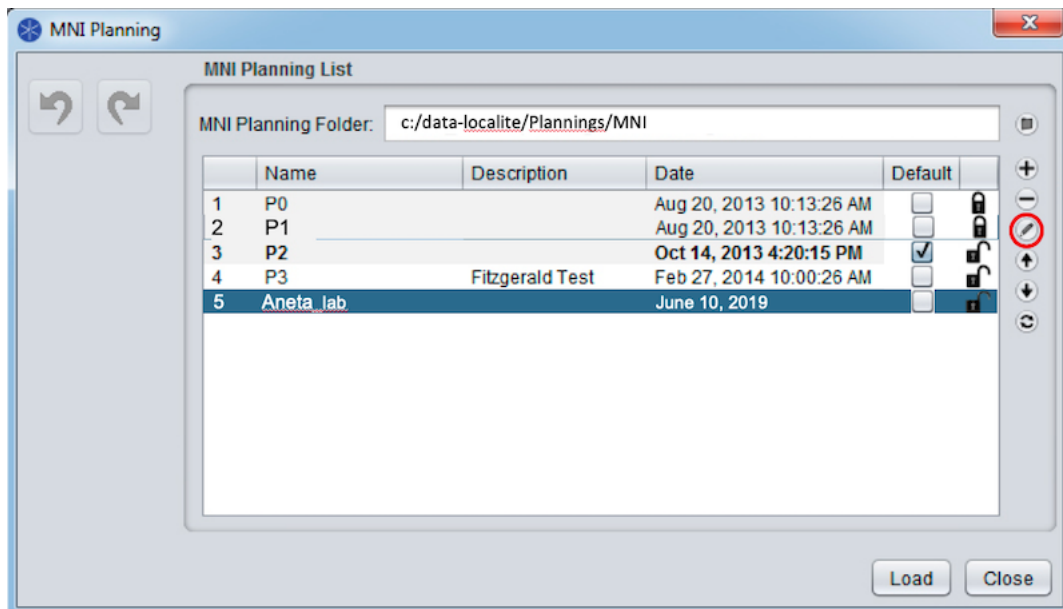
Fig. 12: Here we are creating a new list: Aneta_lab. As soon as we click the **edit** button we can populate the list with our targets specified in standard space.
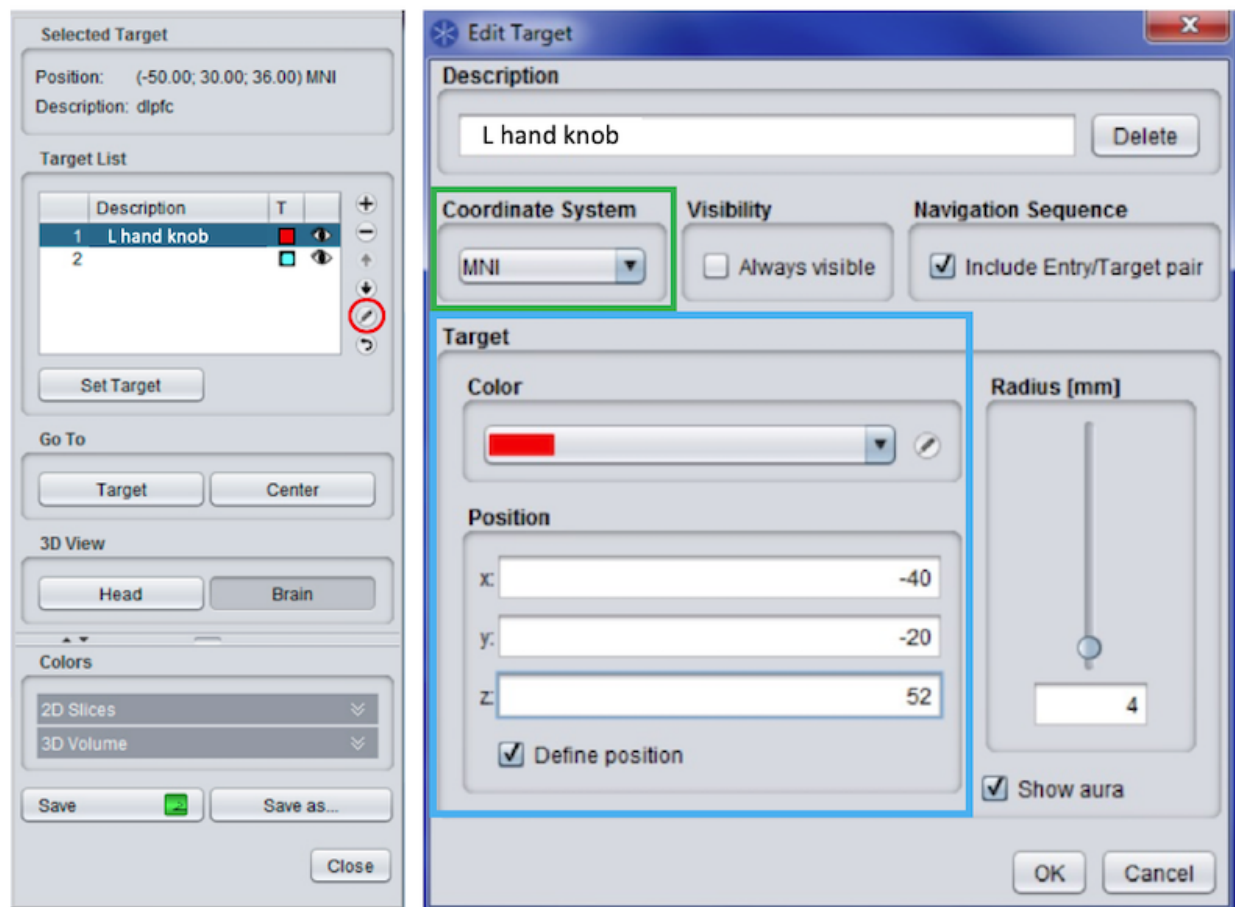
Fig. 13: MNI Planning can save you a lot of time, especially if you need to identify several stimulation targets. **Left**: After selecting the edit button for the MNI Planning list, enter a list of targets using the +. Here we define the left hand knob. You will almost always want either the left or right hand knob defined in order to test for stimulation threshold. Subsequently, edit (pencil in red circle) each target. **Right**: Ensure you are using the desired coordinate system (green rectangle) **before** you set the target coordinates and color (blue rectangle). Press **Save** when you finish defining the targets in your list.

9.6.1 New Data Document
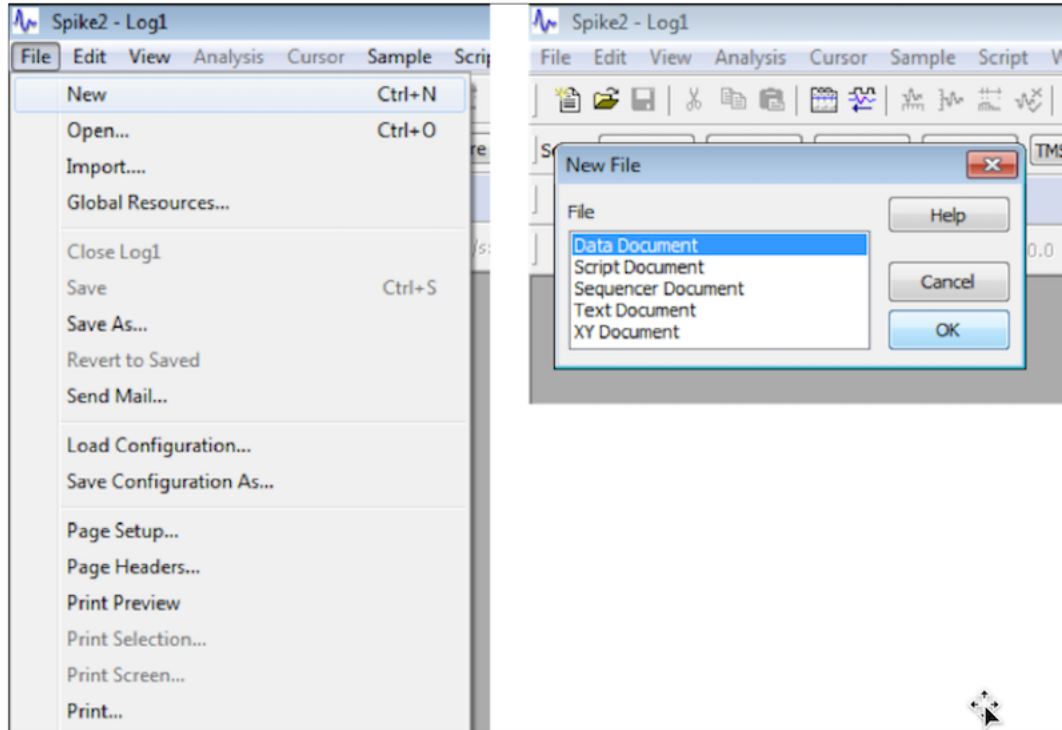
- For each session, create a new Data Document:
- Click the Spike2 icon on the MEP computer desktop.
- File New Data Document OK

9.6.2 Apply Resource File
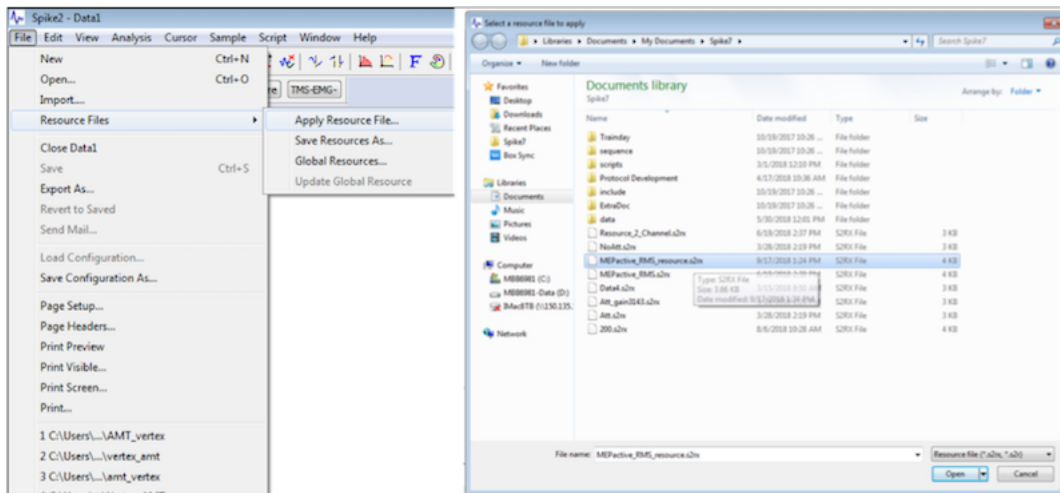
Fig. 14: Apply the resource file: File Resource Files Apply Resource File *MEPactive_RMS_resource.s2rx* (in My Documents\Spike7)

9.6.3 TMS-EMG Configuration
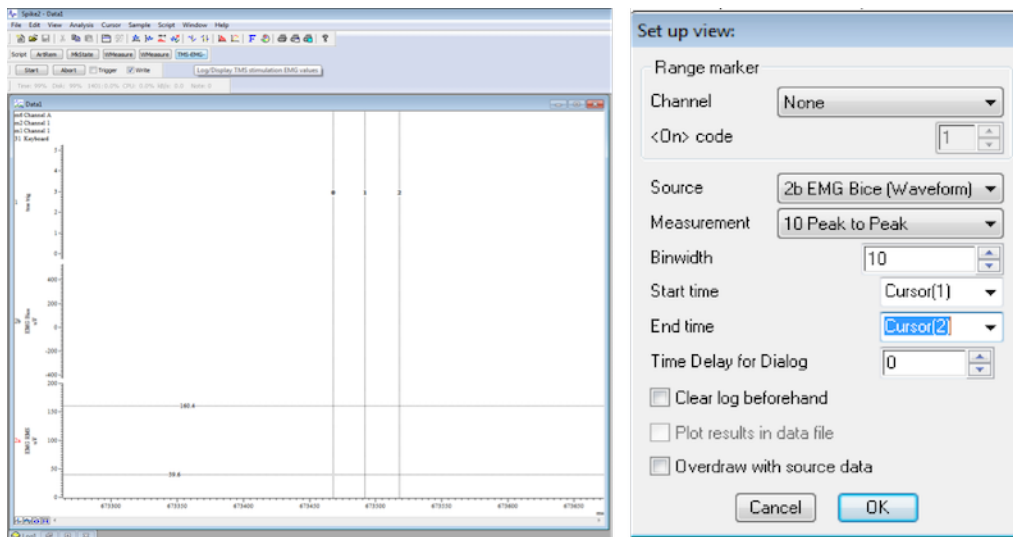
- If the script bar is not visible, then select **Script Script bar** from the menu

- Click View Toolbar (this shows us the *Start* and *Stop* buttons we want later)

- Press TMS-EMG on the Script bar

- Start Time: Cursor 1

- End Time: Cursor 2

- Measurement: 10 Peak to Peak

- OK

Note: Cursor 1 and Cursor 2 may not be visible after hitting okay. Don't worry, they are present but hidden.
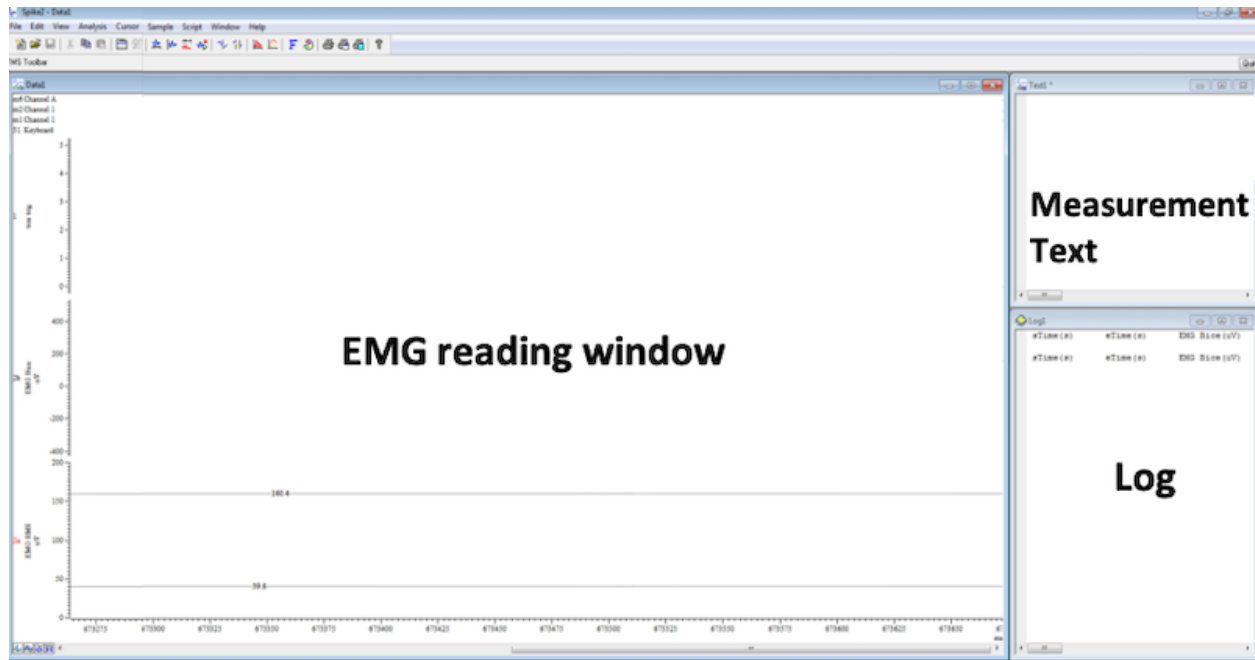
Fig. 15: You should see three windows: the main EMG Reading Window, a Measurement Text window, and a Log.

Create a participant folder. This is where we will eventually save output from the main Spike EMG Reading window and the Log window.

9.7 📄 Set up Session (Localite computer, Toolbar)

- Optimally, you should have a T1-weighted scan with ~1mm isotropic voxels. You have several options for providing an anatomical scan (choose only one option, usually DICOMS):

  1. Bring DICOMS direct from the scanner

  2. Bring a Nifti volume

  3. If your participant is returning for additional sessions, you can load a bookmark from the planning you did last time. From the Menu, choose **Load previous bookmark in New Session**. This will allow you to skip ahead to *participant setup*. By loading the bookmark in a new session, you will have a separate folder for each session, which makes data mangement cleaner.

  4. If you don't have a scan, you can use the standard MNI scan available in TMS-Navigator

- For option 1 TMS Navigator looks for a folder named **DICOMDIR** in your **Patient Folder** and will show you the DICOM images within it.

- For option 2, ensure that a folder containing the NIFTI anatomical file(s) is available.
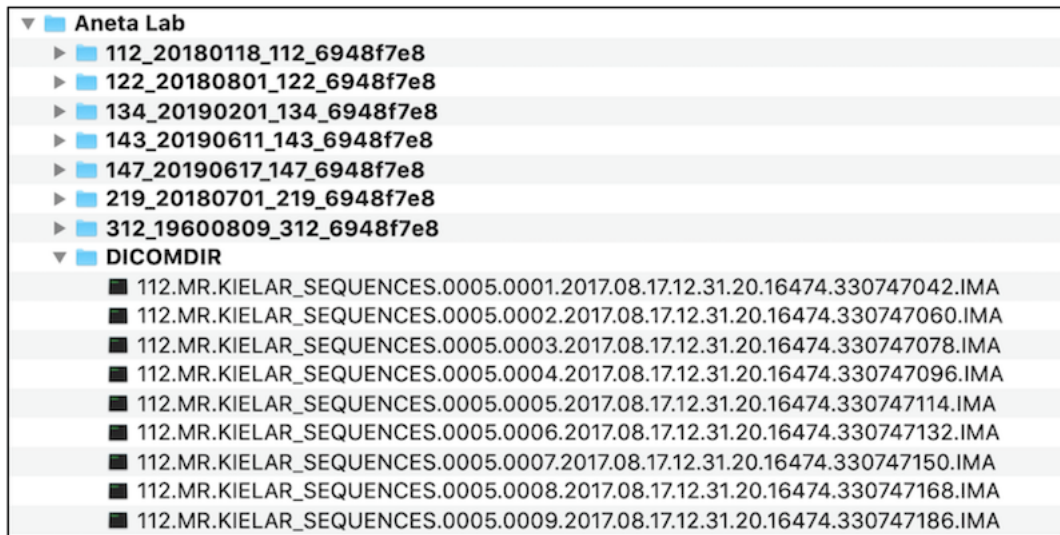
Fig. 16: This is TMS Navigator's preferred directory structure: The **Patient Folder** (called `Aneta Lab` here) is the folder in which you store **all** of your patients, **not the folder for an individual patient**! DICOMDIR (especially relevant if you wish to load DICOMS) is under the Patient Folder and contains all of the DICOM images. **There should be no subdirectories in DICOMDIR!** All of the DICOM images are at the same level.
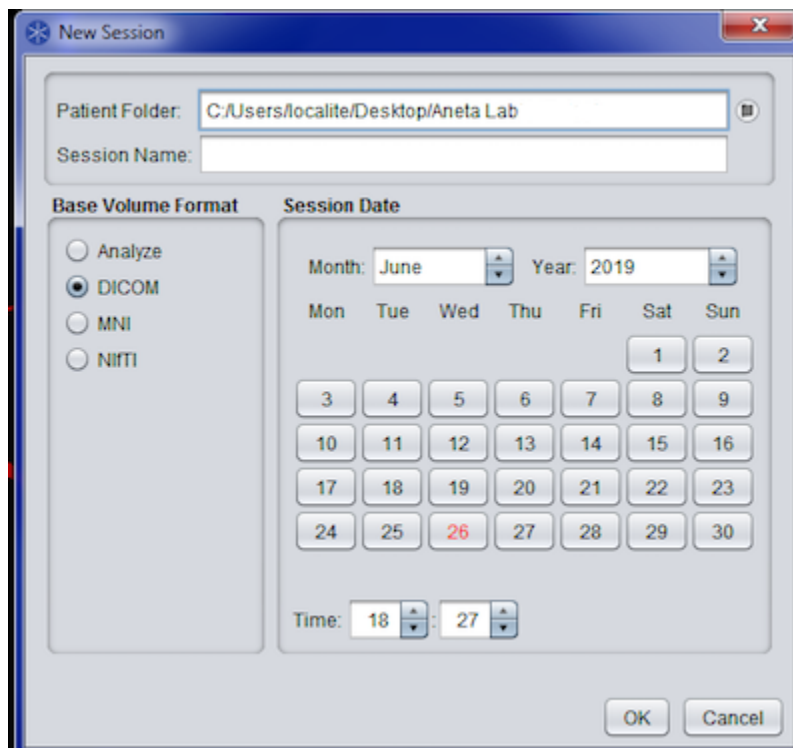
Fig. 17: Ensure you are in the Patient Folder for your study (e.g., `Aneta Lab`). Select the correct Base Volume Format (left). In this figure, we chose `DICOM`. Provide a session name. The individual patient folder gets created in the **Enter Patient Demographics** step described below. At that time, the session will be correctly placed inside the individual patient folder.

9.7.1 Load Anatomical Image: Option 1 DICOMS

In general, you will load anatomical images directly from the scanner. This means they will be in DICOM format. TMS Navigator is specially suited to working with DICOM images.
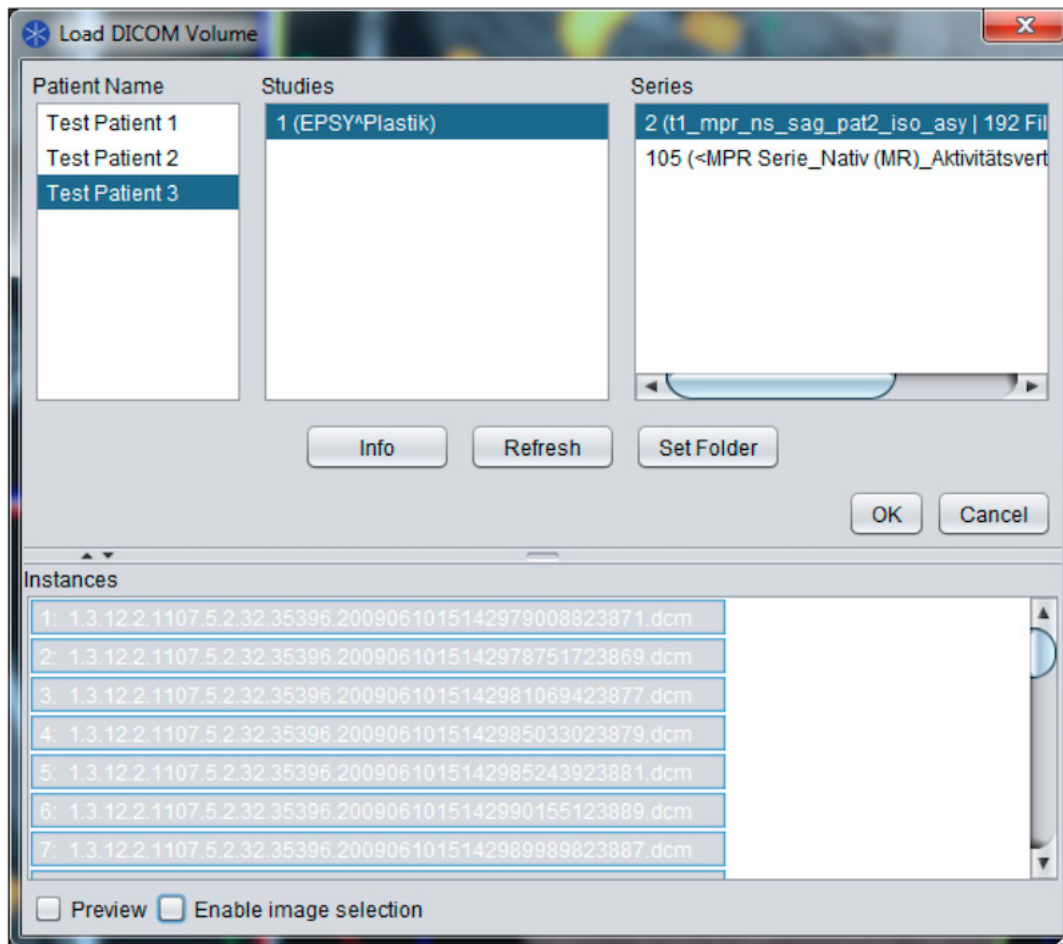
Fig. 18: The **Load DICOM Volume** interface: TMS Navigator looks for the specially named folder **DICOMDIR**, reads the contents and parses the DICOM headers to extract information about the patients, studies and series available. You can view the images by checking **Preview** on the bottom left of the interface. Ensure you are in the correct folder! Do you see your patient numbers? You may need to `Set Folder` to point to your own `DICOMDIR`.

9.7.2 Load Anatomical Image: Option 2 NIFTI

9.7.3 Enter Patient Demographics

Regardless of whether you loaded DICOMS or NIFTIs, you need to check that your patient demographics information is correct.
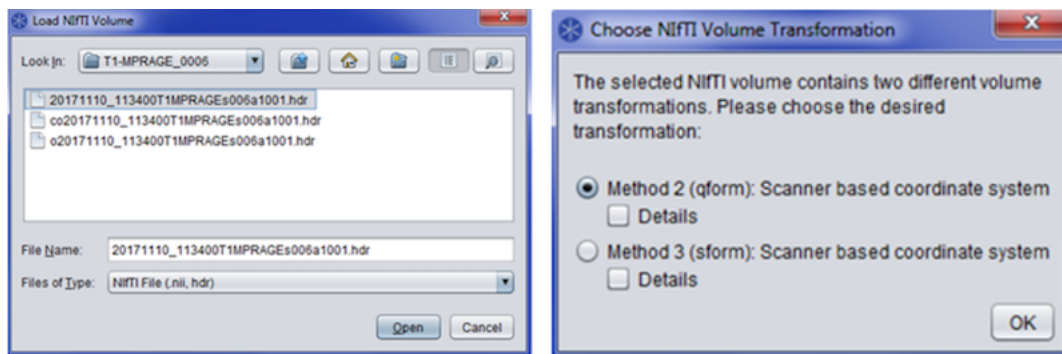
Fig. 19: **Left**: If you load a NIfTI volume, you must choose the NIfTI file (*.nii or *.hdr) and open it from a location of your choosing. **Right**: For a Nifti file, you must specify a NIfTI Volume Transformation: Method 2 (qform) → OK. The Method (qform or sform) is specified in the TMS Navigator preferences, and is currently set to qform.
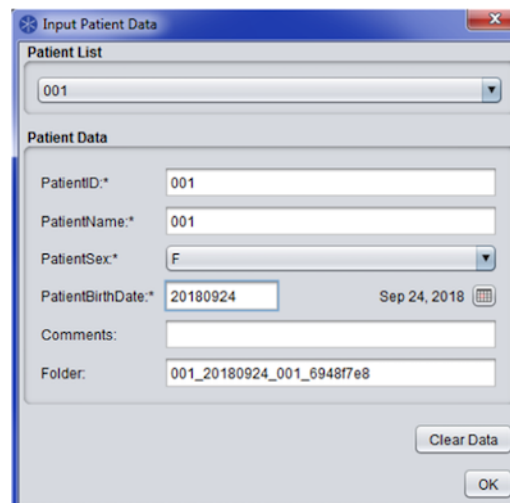
Fig. 20: Do one of the following: 1) Select an existing patient from the **Patient List** (top) or 2) Enter new patient data: birthdate=today, name=ID; Sex. OK. If this is a new patient, TMS Navigator creates a folder for the individual patient based on the **PatientID**, **PatientBirthDate**, **PatientName** and the software's **Application Instance UID** (see `Folder` at the bottom of the figure). The individual patient folder name gets added to PatientList.xml in the Patient Folder (e.g., `Aneta Lab`) and will become available in the **Patient List** next time. If the patient already exists, then the new session will be added to that patient's folder.

> **Warning:** If you do not select a patient from the patient list, TMS Navigator will create a second individual patient folder! This is true even if the patient already exists and even though the correct information is displayed in the **Patient Data** section! **So, make sure you choose your patient from the patient list at the top (unless it is really a new patient).**

9.7.4 Surface Threshold
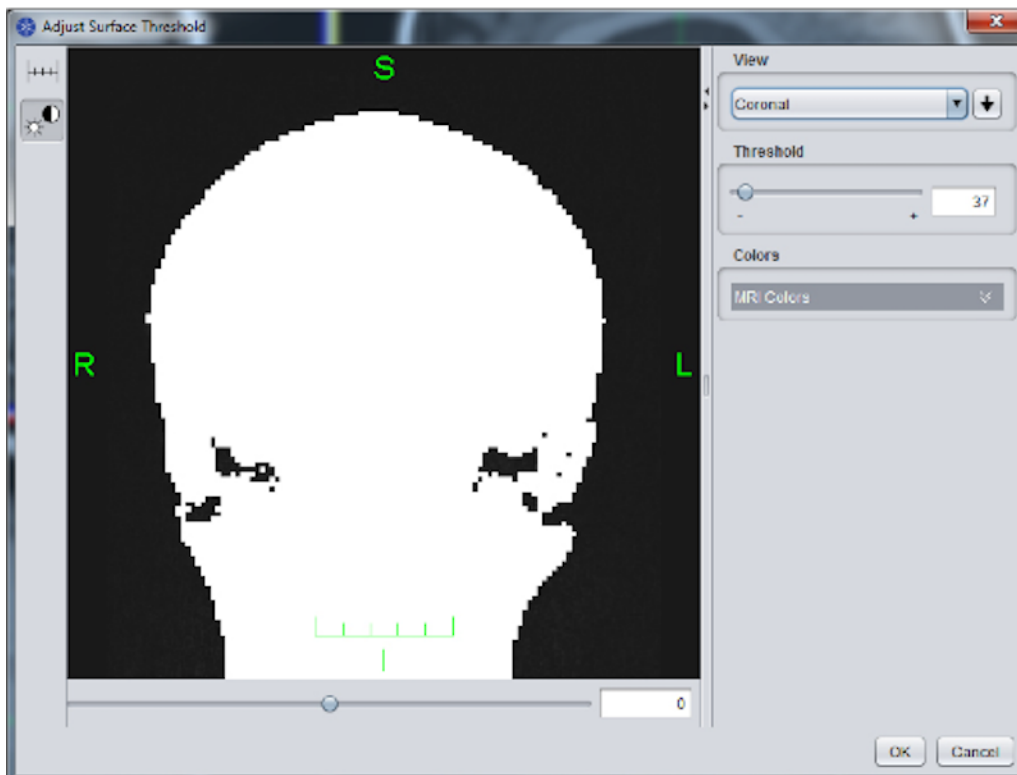
Fig. 21: Adjust Surface Threshold (usually ~50) with the threshold slider. Check that whole brain is included in each view plane. Spots outside the head are preferable to holes in the head. Click **OK**. TMS Navigator needs an accurate estimation of where the surface of the head is relative to the brain inside.

9.8 Patient Registration Landmark Setup (Toolbar)

- In addition to surface registration, specific landmarks help the software track location accurately
- Define Registration Markers (upper right in figure below): Marker List + for each new marker:
- (1) L_ear, (2) L_eye, (3) Nasion, (4) R_eye, and (5) R_ear
- Double check colors and locations.
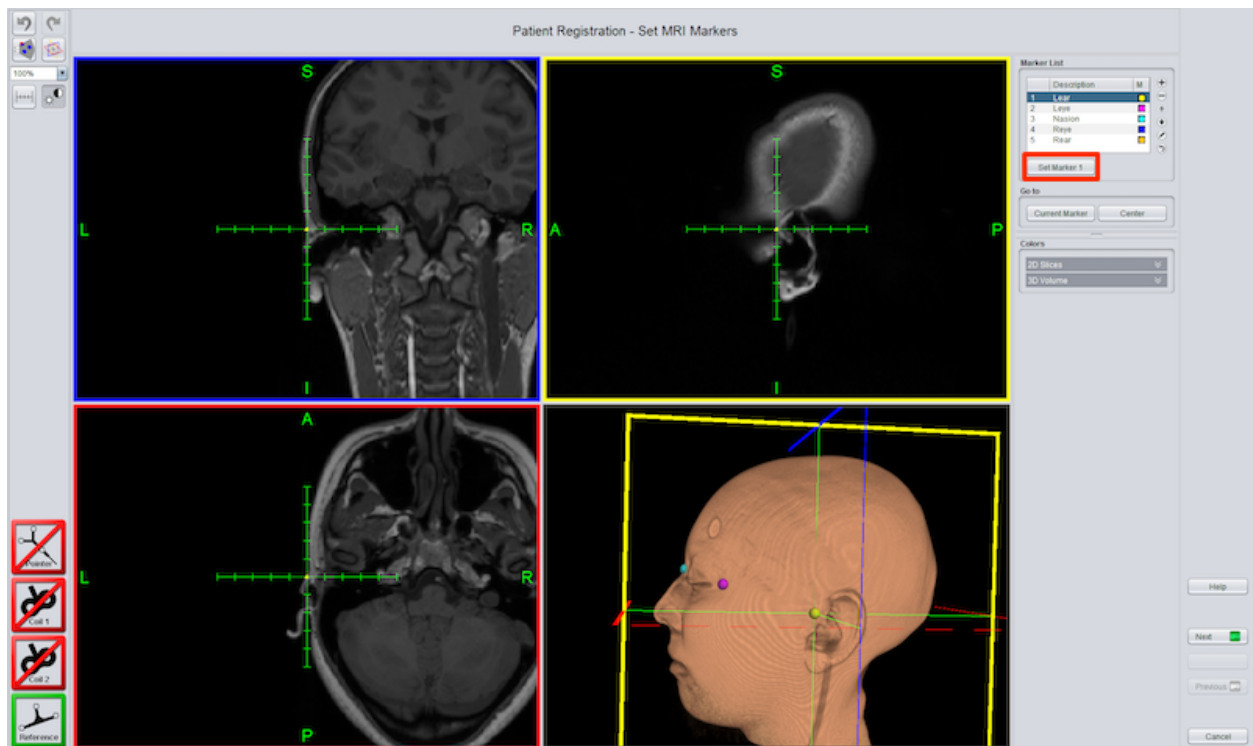
Fig. 22: Markers and their colors have been defined in the Marker list (upper right). Each marker (colored ball) has been set to a location on the head. **Moving the cursor**: Either drag one of the 2D views under the cursor or double-click a tickmark on the cursor to move to that location. Here the left ear marker is selected in the list and display. The marker could be moved by selecting a different location in one of the 2D views and clicking *Set Marker 1* (red rectangle).

- **Cancel** (Because the participant is not here to be registered).

9.9 ![brain icon] Brain Segmentation (Toolbar)

- Brain segmentation creates a 3D image of the brain which is useful for placing the grid correctly during Talairach definition. The brain segmentation also provides us with a nice view of the sulci and gyri of the brain which facilitates locationg targets.
- Click the cursor in the middle of a large region of WM Calculate (red rectangle).
- If the cerebellum and/or spinal cord is not included, click each and calculate again.
- Try Defaults, else Range=7-34; Sensitivity=12.
- Increased range Increased coverage
- You can reset segmentation and start over.
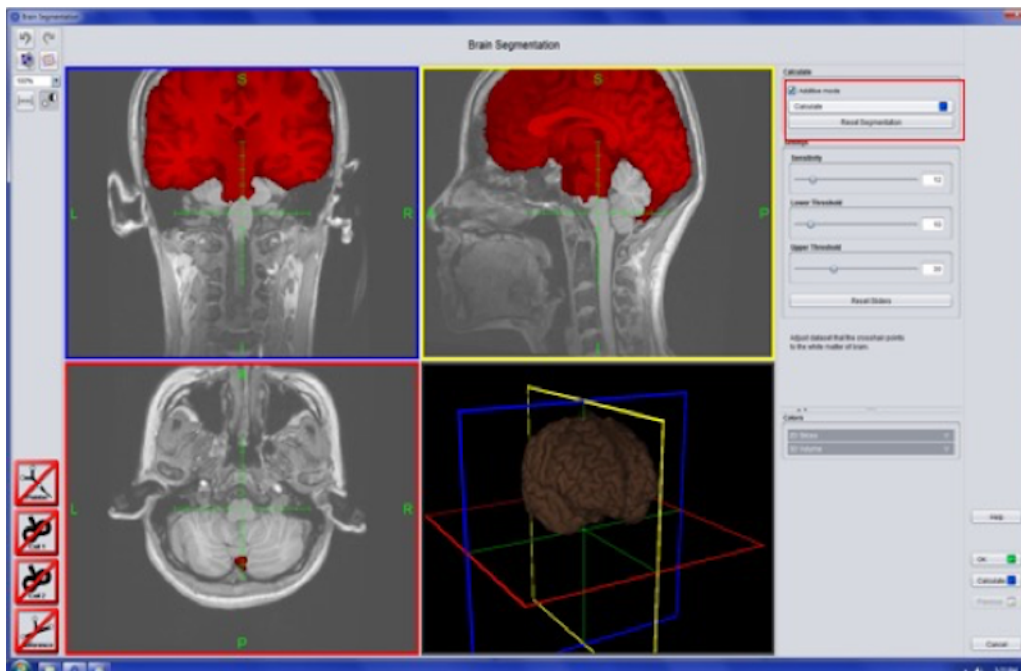
Fig. 23: The cerebellum was not successfully segmented in this image. To include the cerebellum in the red mask, click the cerebellum and choose *calculate* (upper right) again.

9.9.1 ![MRI icon] Brightness and Contrast (Lower Control Area)

Adjust window width (contrast) and window center:

- Reduced width (range) increased contrast
- Decreased center increased brightness

9.9.2 ![icon] Peel surface (Lower Control Area)

If gyri/sulci are cloudy then peel the brain surface (~3mm)

9.10 Define Talairach (Menu)

We define Talairach coordinates so that locations in the individual brain can be located in a standard coordinate system. Defining Talairach for each participant is crucial to using the MNI planning that we set up earlier.

On the menu:

- File Talairach Definition Setting markers: *Comissura anterior*, *Comissura posterior*,
- *Point of Falx cerebri* OK (for falx: Align with AC, Selected location defines top of brain).
- Next
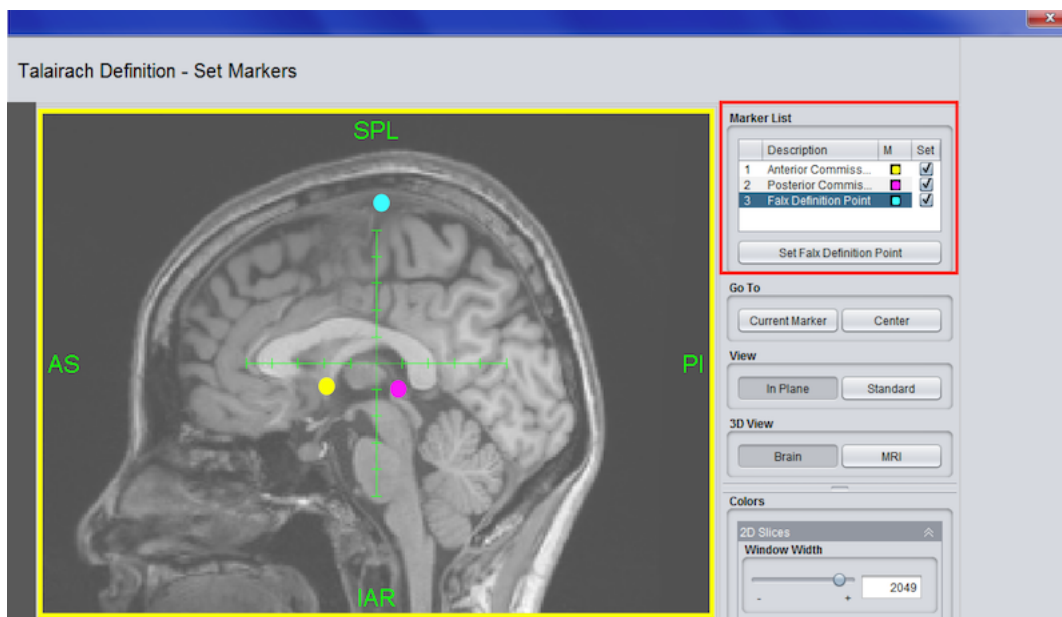
Fig. 24: A sagittal view of the three points you need to find for talairaching: Anterior Commisure (AC) in yellow; Posterior Commisure (PC) in pink; Falx Cerebri (defines top edge of brain tissue at the midline) in cyan.

Note: Although we use a simple talairaching definition to estimate standard coordinates, we can freely choose either MNI or Talairach coordinates when we plan targets. All we have to do is **ensure we have explicitly selected which coordinate system we are using when we define the MNI Planning**. This website can help you translate between MNI and Talairach coordinates if necessary.
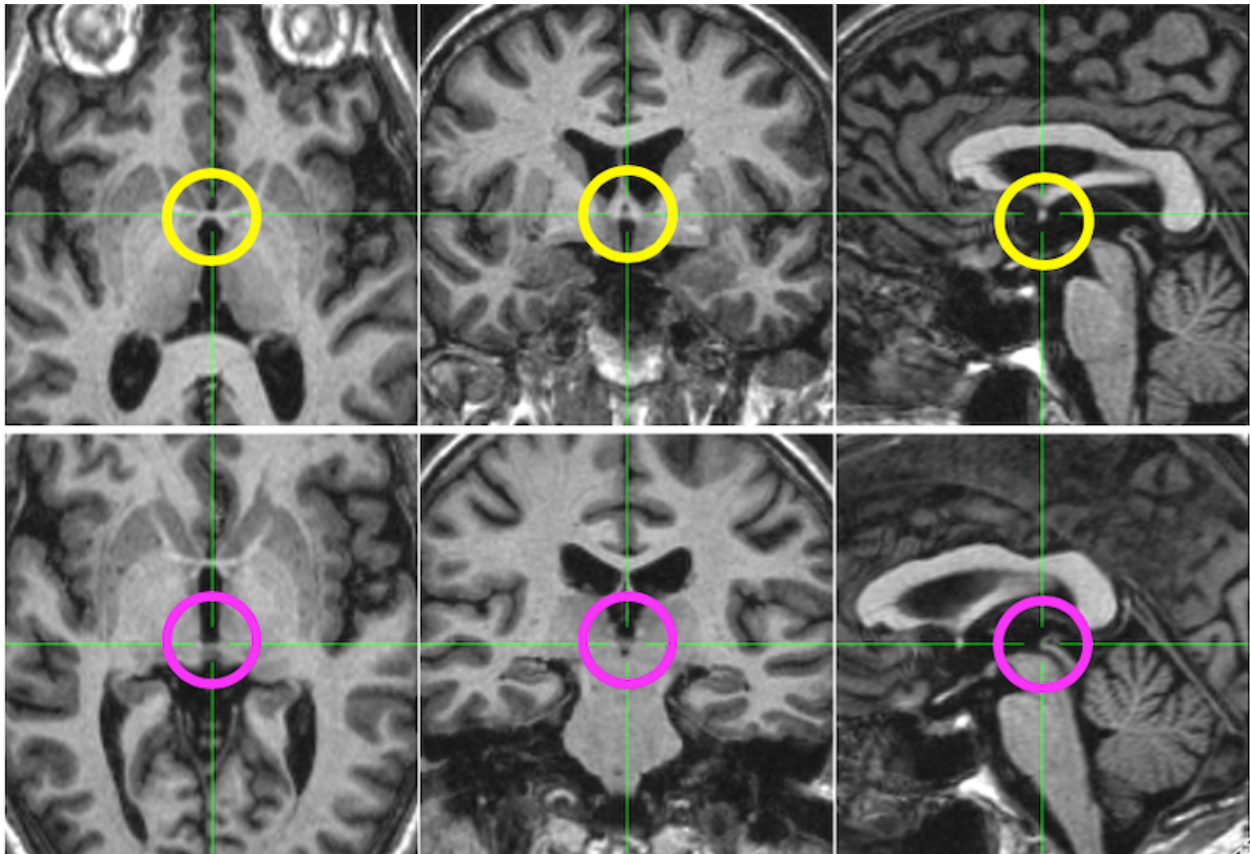
Fig. 25: Top: The **anterior commisure** (AC) is displayed in the yellow circle. It is a little bridge of tissue between hemispheres. The sagittal view displays a typical cross-section of the AC. Bottom: The **posterior commisure** (PC) is displayed in the pink circle. It is also a little bridge of tissue between hemispheres, but it is typically more difficult to locate than the AC. You probably need multiple views to locate the PC.
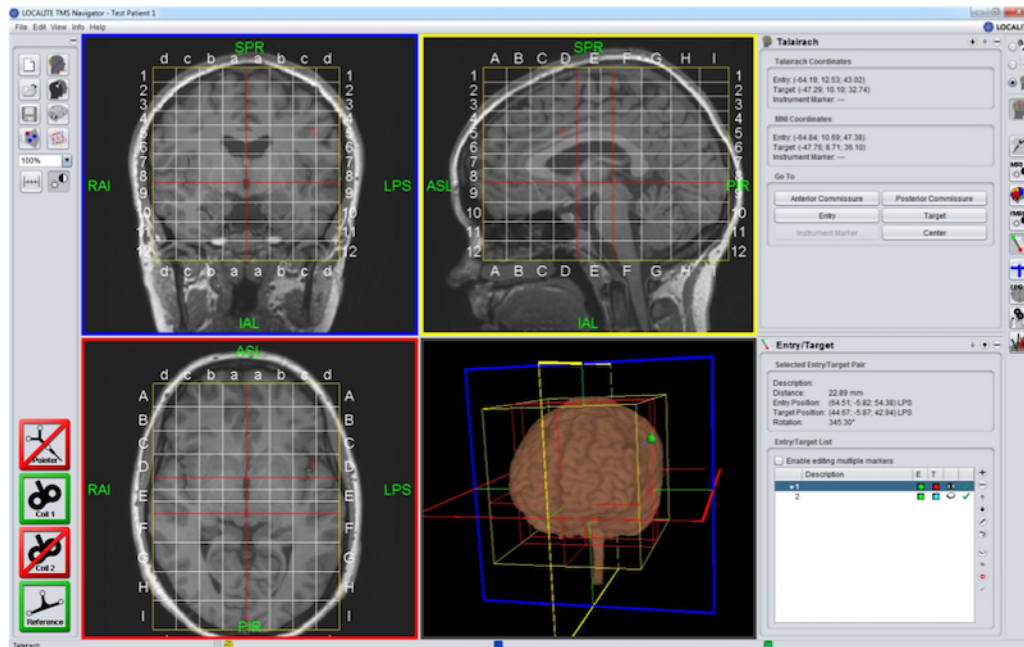
Fig. 26: Drag edges of cage to tightly encompass whole brain+cerebellum. Use 3D image to make sure no brain is outside the cage grid. OK.

9.11 Load MNI Planning Targets

Now that the participant's brain is registered to standard space, you can load a saved MNI Planning list, e.g., File MNI Planning Kielar lab Load (Load standard space targets)

9.12 Rotation and Entry for each Target

For each target defined with MNI planning, we need to know the corresponding location for stimulation on the surface of the head.

In the lower control panel:

- Click the Entry/Target icon to view a list of your loaded targets.

- Click the Planning icon to **Define Rotation** and **Calculate Entry**. The entry is the location on the head where the coil must be applied to stimulate the desired target.

In the Planning Control Panel:

- Check **Define rotation**.

- Set rotation to 45 degrees in the left hemisphere and 315 degrees in the right hemiphere.

- Click **Calculate Entry** in the Planning panel.

- If you have reason to set a different rotation value, you certainly should, but the above values are typical.
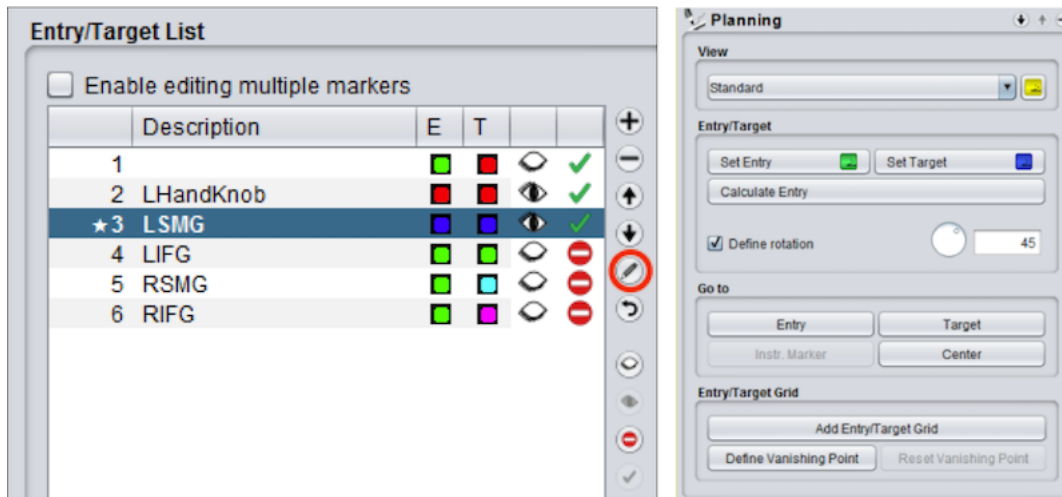
Fig. 27: **Left**: E=Entry; T=Target. Click color squares to set colors (you may want the entry and target to be the same color). − removes an entry; + adds an entry. An open eye shows the Entry/Target in the display. **Right**: For each target in the Entry/Target List (but see *Hand Knob*), set the rotation and calculate the entry.

9.12.1 The Hand Knob

9.13 Set up Participant

- **Paperwork** Explain the study, have the participant read and sign any consent forms, and fill out the MRI and TMS safety screening forms. Double check for contraindications!

- Provide **earplugs** to the participant to protect their hearing from the noise of the coil.

- Place a **reference tracker** on the participant's forehead. The reference tracker is available in two forms: a stick-on version, and a headband version. In general, the **stick-on reference tracker is to be preferred** as it is better tolerated and more reliable. Depending on the stimulation site, you might have to use the headband (if it is not possible to place reference on forehead). If using the headband, make sure it is not too tight (can cause headaches) and not too loose (can cause the reference tracker to shift).

9.13.1 Electrodes

Electrodes are placed on the participant's hand to establish the motor threshold. Choose the correct hand (e.g., the right hand if we are stimulating the left hemisphere.

Note: No matter the selected muscle, place electrodes as follows: red = muscle belly, white = tendon/bone (lateral face of selected finger), black = bony protuberance of ulnar head for grounding. Before placing the electrodes, you can ask the participant to flex the targeted muscle while palpating the muscle belly. You should feel the muscle contract (contracted muscle = firm), which will inform optimal placement of the muscle belly electrode. For the APB, illustrated here, have the participant repeatedly touch the thumb to the pinky to find the muscle belly.

- Demonstrate the pulse at lower intensity (30) on participant's arm (as long as there is no metal in the arm) and then the head (45).
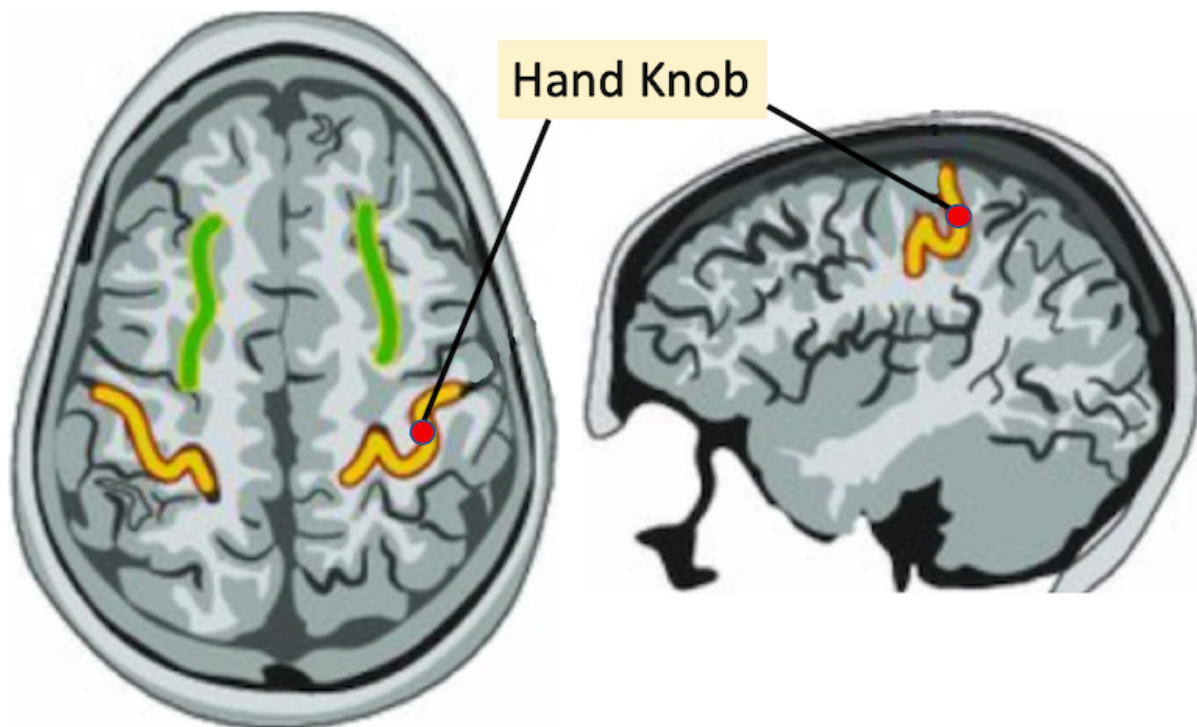
- Ask them to relax their hand.

Fig. 28: It is likely that the hand knob will not be located correctly, though it should be close. Choose the hand knob in your entry target list and then double-click the correct location on the axial display (**red dot** on the omicron-shaped gyrus) using the guide in the figure above. Choose **Set Target** and ensure that you define rotation and calculate entry in the *Planning Control Panel*.
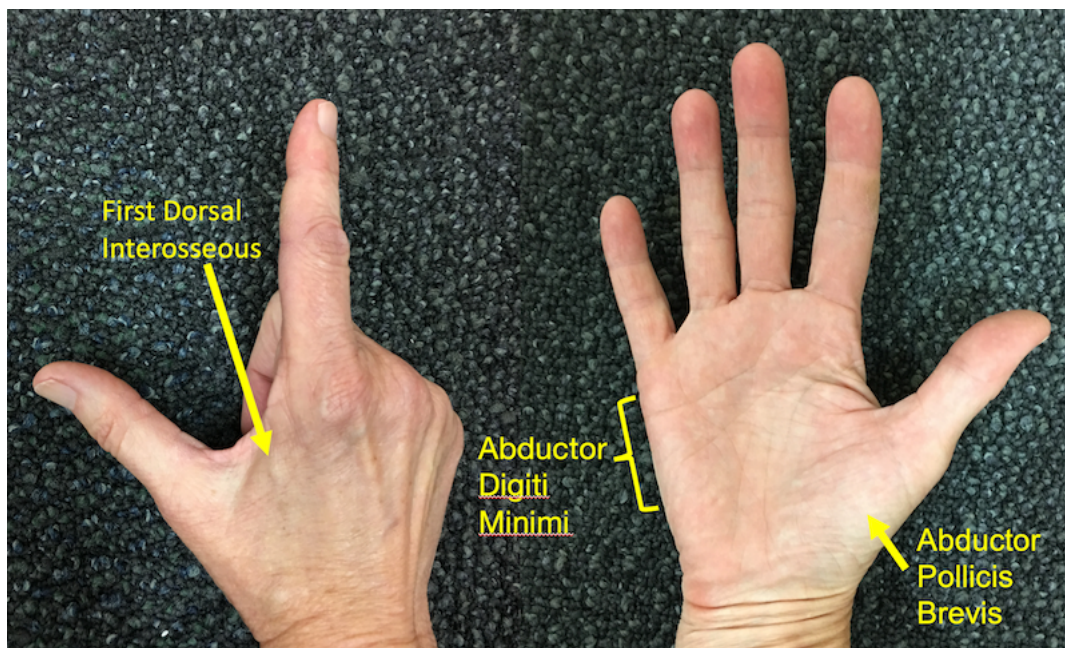

Fig. 29: Common muscles to choose are the FDI (First Dorsal Interosseous), the ADM (Abductor Digiti Minimi), and the APB (Abductor Pollicis Brevis);

Fig. 30: Electrode placement illustrated for the APB: Red EMG electrode thumb's muscle belly (Abductor Pollicis Brevis); White EMG electrode thumb; Black EMG electrode back of wrist (Near Ulnar Styloid) for grounding.

9.13.2 Resume Patient Registration

Now that the participant is here and has a reference tracker affixed, we need to yoke the real world landmarks and head surface to the MRI data in TMS Navigator.

Note: The display contains four equal sized panels. To change the layout, choose *View Layout* in the menu. For example, you can maximize the size of panel 4 to make it easier to see the 3D representation.

Acquire Landmarks

Point to each landmark on the participant with the pointer (guide the pointer with your finger, covering the tip to prevent slipping/unwanted contact). The pointer location needs to match the landmarks on the MRI scan. For example, the specified landmark location for the left ear may vary by operator (always aim for left tragus). Ensure consistency between the marked target on the MRI scan and pointer placement. Use the 3D head image as a guide.

Surface Registration

- Place the pointer on head Blue footpedal down Drag Blue footpedal up pointer off head.
- Repeat in several directions, left & right (100-300 points).
- Get extra registration markers over the eventual stimulation site.
- Be careful not to push too hard as this can cause discomfort. You want to keep contact with skin, but just lightly tracing the pointer rather than pressing firmly.
- Click **Next** (green button) when you are finished adding points to recalculate the RMS.
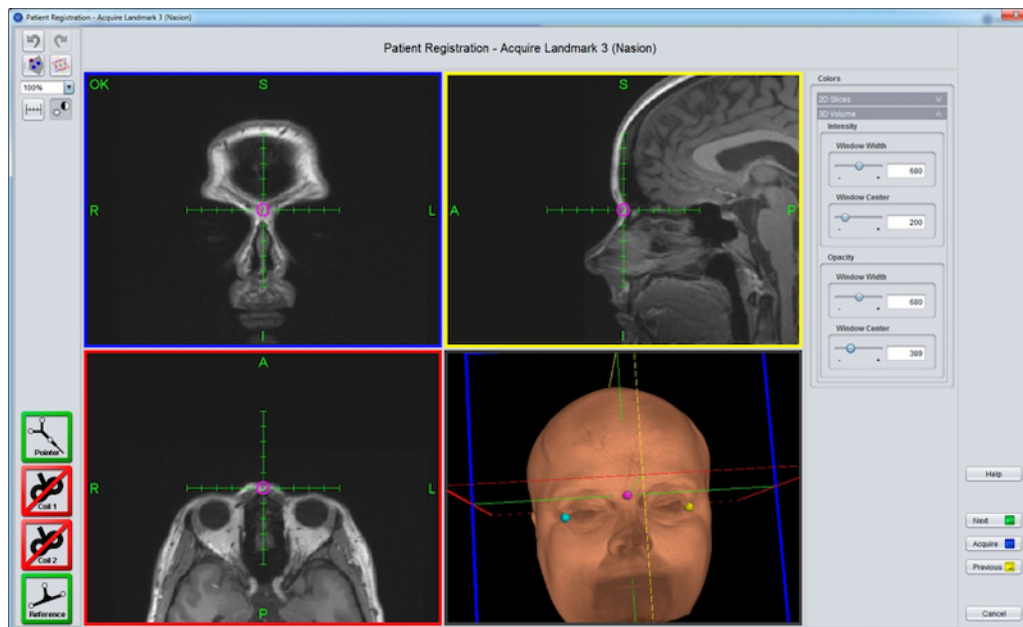- To accept the new RMS, click **OK** .

Fig. 31: Acquiring landmarks is only successful if both the **pointer** and the **reference** (on the participant's forehead) are visible (green) in the Lower Toolbar as illustrated here. Step on the blue footpedal to **Acquire** each landmark, or ask the computer operator to press the **Acquire** button (blue button, lower right). After acquiring all five landmarks, press the **Next** button (green button, lower right).
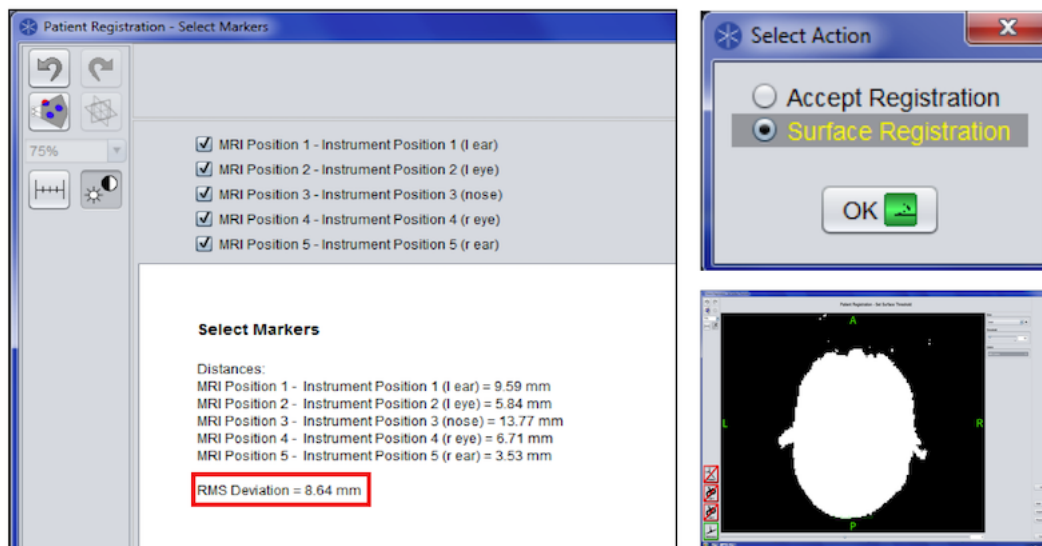

Fig. 32: **Left**: TMS-Navigator evaluates the registration quality by providing a RMS (Root Mean Square) deviation value (red rectangle). In this case, the RMS is 8.64, which is too high. **Upper Right**: Select **Surface Registration** to try to improve the RMS. OK. **Lower Right**: We are presented with a surface threshold. It is probably fine, but check it.

- To reject the new RMS, click **Previous**, **Next** and **No**. This should allow you to add more data to the surface registration.

- Hold the pointer so it is touching the head and perpendicular to it. You should see no gap between the pointer and the surface of the head (2D views).
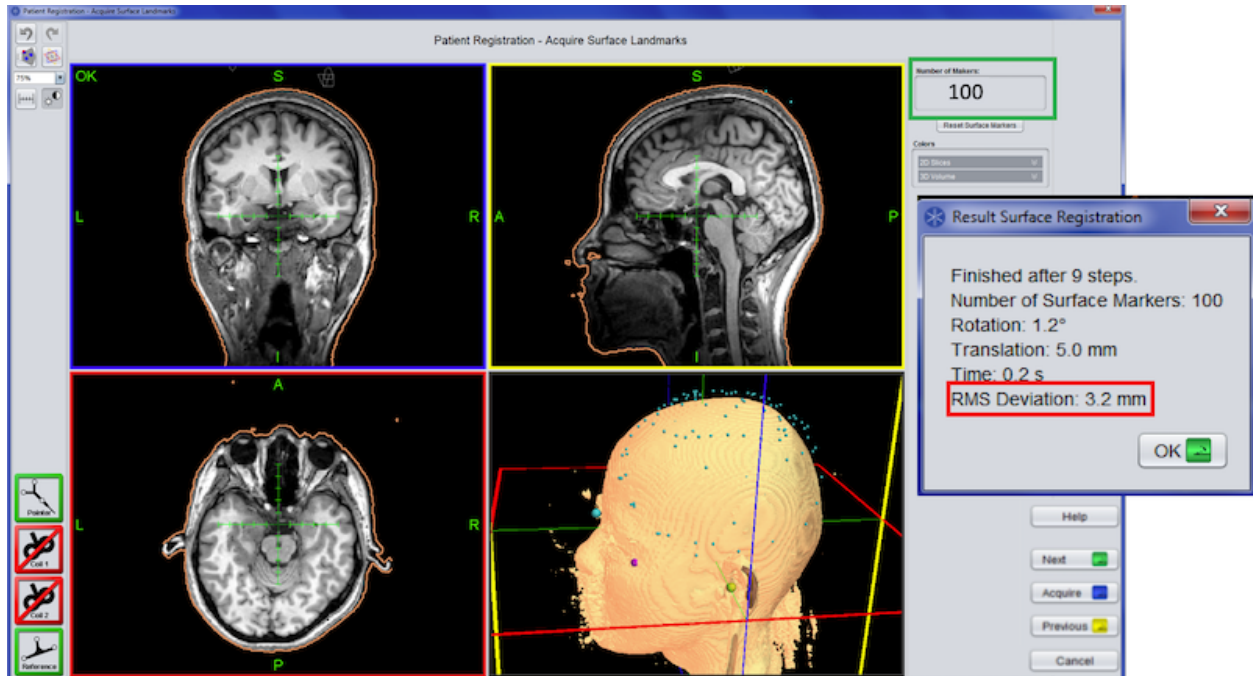
Fig. 33: Here we have 100 points (green rectangle, upper right). We have clicked **Next** (green button, lower right) to recalculate the RMS. **Right**: Our RMS is now 3.2 mm. Click the green **OK** button on the **Result Surface Registration** window to accept.

9.14 Coil Calibration

Calibrate the coil to ensure it is recognized by the camera.

 Select the **Instrument** icon (Lower Control Area)
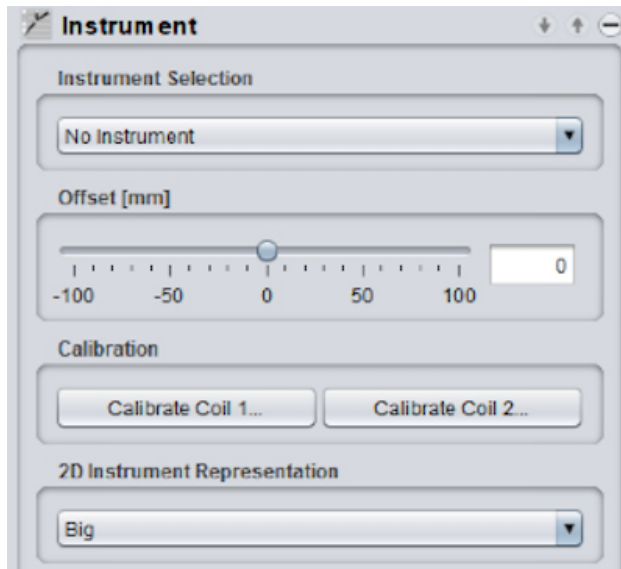
- Instrument selection Coil 1 (MagVenture C-B60)

- Place the calibration board on the coil and hold it near the participant.

- Tip the board and coil so the camera can see all the reflector balls.

- Calibrate Coil 1…
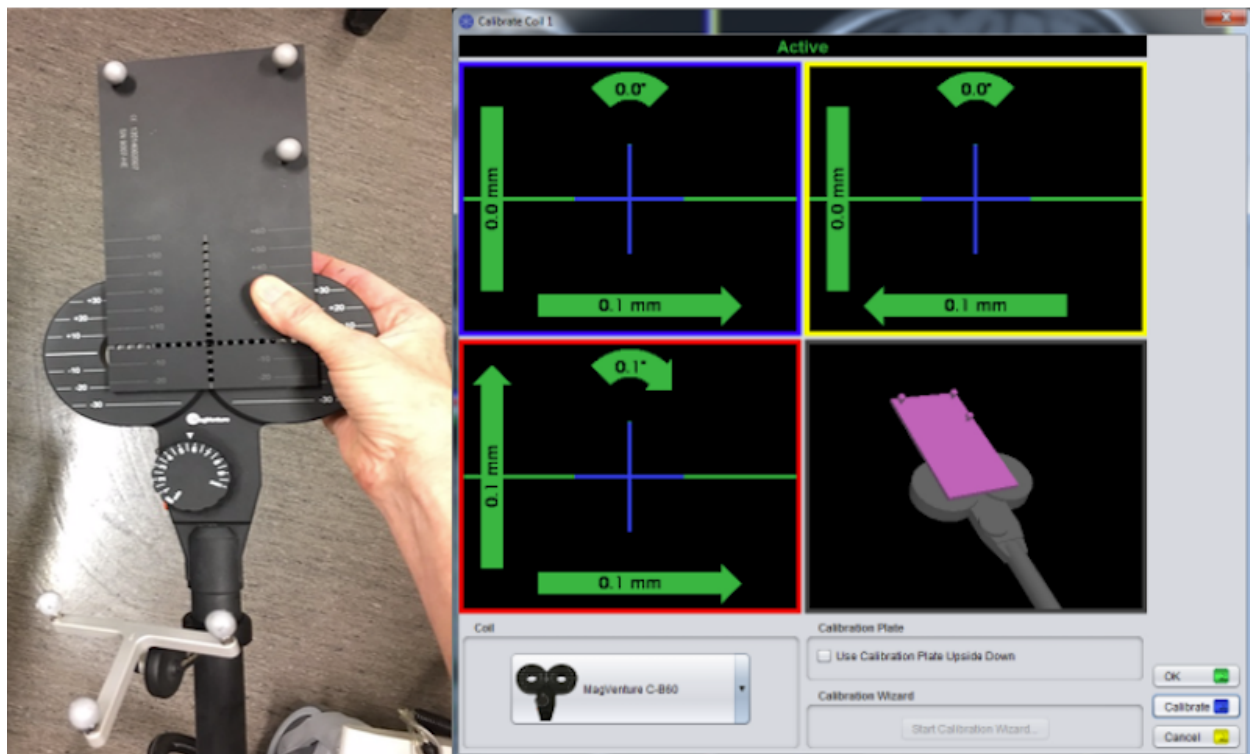
Fig. 34: **Left**: C-B60 Coil with aligned calibration board. **Right**: When the instrument calibration displays **Active** in green at the top of the screen, click the blue **Calibrate** button, then **OK**. The coil is now calibrated.

9.15 Check Camera (Polaris Spectra P7) Position (Toolbar)
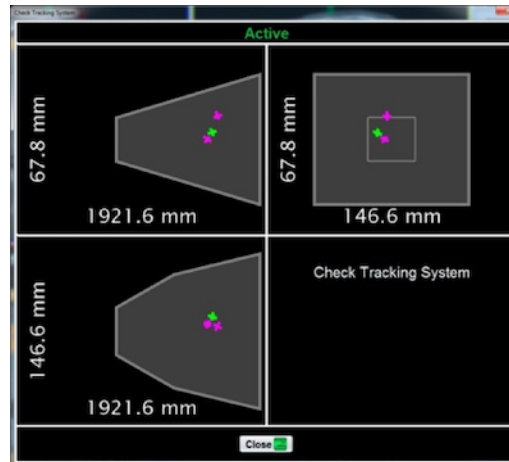
Fig. 35: If the Reference icon is red, the reflector balls on the participant's head are not being detected, and you should check the camera position. If detected: pointer = green cross, and the reference tracker = pink cross. Coil visibility can be checked as well (pink crosses). Undetected balls are red dots.

9.16 Set up to Find Motor Hotspot

- We have set up the equipment and the participant. We have planned our targets and entries.
- We are going to stimulate the hand knob, and subsequently revise the location of the hand knob using an instrument marker from successful stimulation.
- We'll need the four control panels described below.
- Switch to Navigation Mode (Upper Control Area)

9.16.1 Stimulation (Lower Control Area)

Note: If you forget to choose **Update during recording** never fear. The stimulation markers are still being created, you just won't see them populate the list right away. But, you should see them when you press **Stop**.
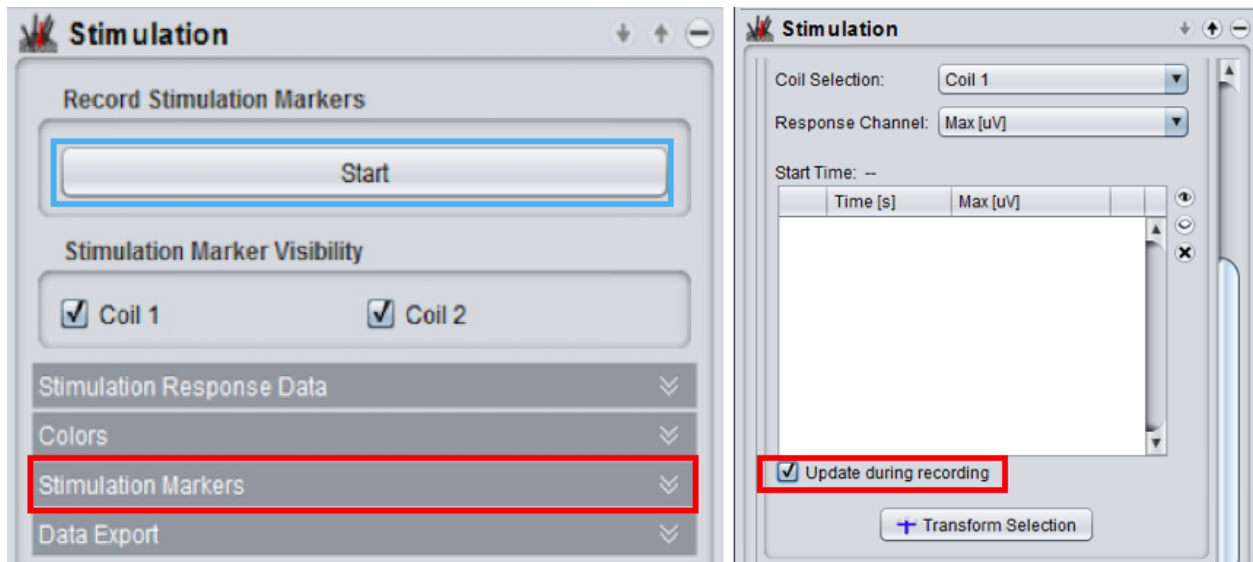
Fig. 36: **Left**: The stimulation control panel. Click **Start** (blue rectangle) [The word **Stop** replaces **Start** on the button. We'll use that later.] Open the **Stimulation Markers** dropdown (red rectangle). **Right**: The empty stimulation markers list is displayed. At the top, coil 1 (the small coil C-B60) should be selected. Check **Update during recording** (red rectangle).
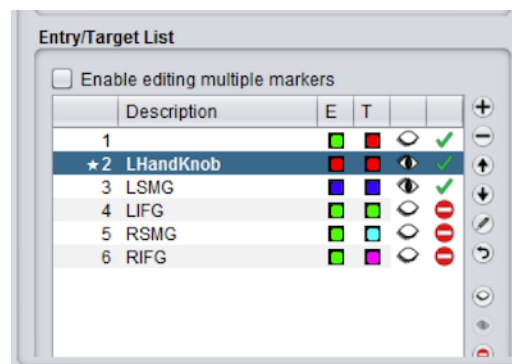

Fig. 37: Select the hand knob from the list.

9.16.2 Choose Entry/Target (Lower Control Area)

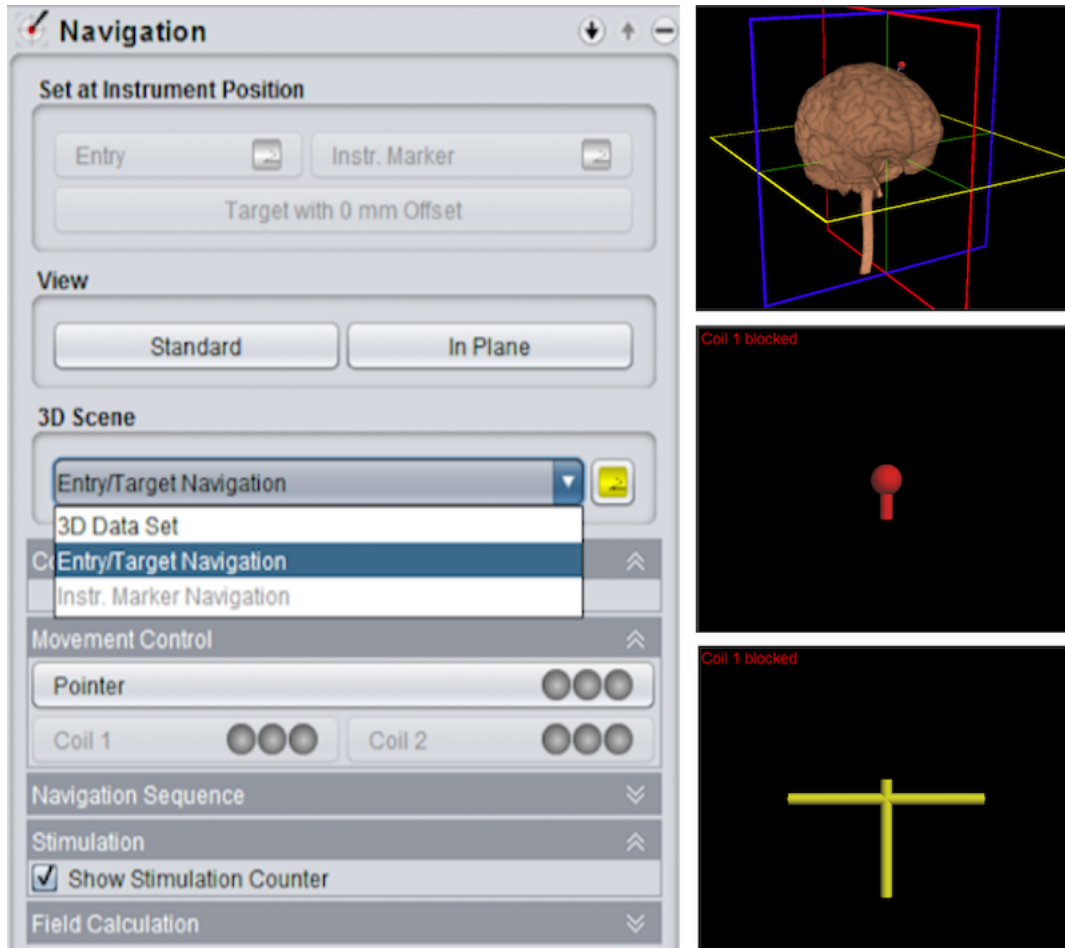## 9.16.3 Set up Navigation (Lower Control Area)

Fig. 38: **Left**: The Navigation control panel showing the 3D scene dropdown. You can select from the dropdown or cycle through with the associated yellow button. **Right**: From top to bottom: 3D Data Set view, Entry/Target Navigation view and Instrument Marker view. Choose **Entry/Target Navigation**.

9.16.4 Stimulator

9.16.5 MagPro Settings

On the MagPro screen, ensure the coil is correctly identified as the C-B60 (the small coil). Turn the Options Wheel to A. Select **Recall**. After selecting **Recall**, the details of your sequence are displayed in the information area on the
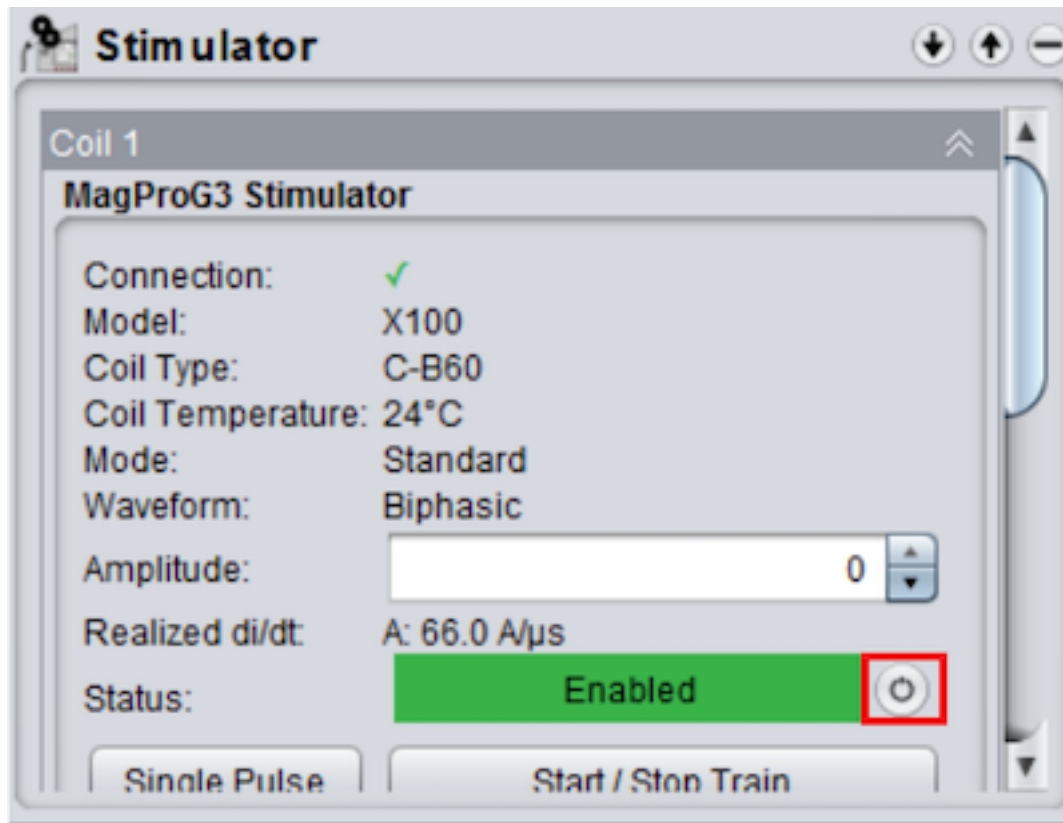

Fig. 39: Stimulator Control Panel: This panel allows you to (1) enable a coil, (2) set the coil stimulation amplitude, and (3) start recording from the coil. To **enable a coil**, change its status Disabled→Enabled by clicking the button to the right (in the red box). You can set the stimulation amplitude in the **Amplitude** textbox. If you use the arrows to set the stimulator amplitude, then the settings will be applied immediately. If you type an amplitude into the box, then you need to hit enter to apply the settings. You will need this when you stimulate the hand knob with various amplitude values dictated by *Pest*. The coil operator can trigger a pulse with the orange button on the coil.

left: `Setup View A`. You may use the **Amplitude Wheel** and **Enable/Disable** button if you prefer these to the Stimulator panel.

> **Warning:** When you enable the coil, amplitude is reset to `0`! You will need to set amplitude to `55` when you try to locate the motor hotspot. If you switch between different methods of setting the amplitude (dial on C-B60 coil, MagPro amplitude wheel and stimulator control panel), the amplitude may jump when you switch back to the previous method. Be cautious about this.

9.17 The Motor Hotspot

- Locating the motor hotspot requires TMS-Navigator on the Localite computer and Spike 2 on the MEP computer.

- Establishing the RMT (Resting Motor Threshold) additionally requires Pest on the MEP machine.

9.17.1 Start Sampling with Spike 2

 Sampling Speed is rather fast, click the slow button in the lower left corner of the Spike2 interface. About 4 clicks should do it.

 If you go too far, there is an adjacent icon for speeding up the sampling.

Adjust the Y-axis (Spike 2)

9.17.2 Locate the Motor Hotspot

- **Set the amplitude** to `30`. The amplitude can be set in at least 3 different ways:

  - Using the dial on the C-B60 coil

  - Using the amplitude wheel on the Magpro

  - Using the Stimulator control panel in TMS Navigator

- Demonstrate the coil stimulation on the participant's arm at amplitude= `30` (as long as there is no metal in the arm), and head at amplitude= `45`.

- Set the amplitude= `55`, then navigate to the handknob and stimulate it. You may have to move the coil around to find the correct spot.

- Watch the participant's hand. The thumb, and not some other part of the hand, should move.

- Often this is right where we placed the handknob target, but if that isn't right, we need to move the coil around a bit and try stimulating nearby areas.
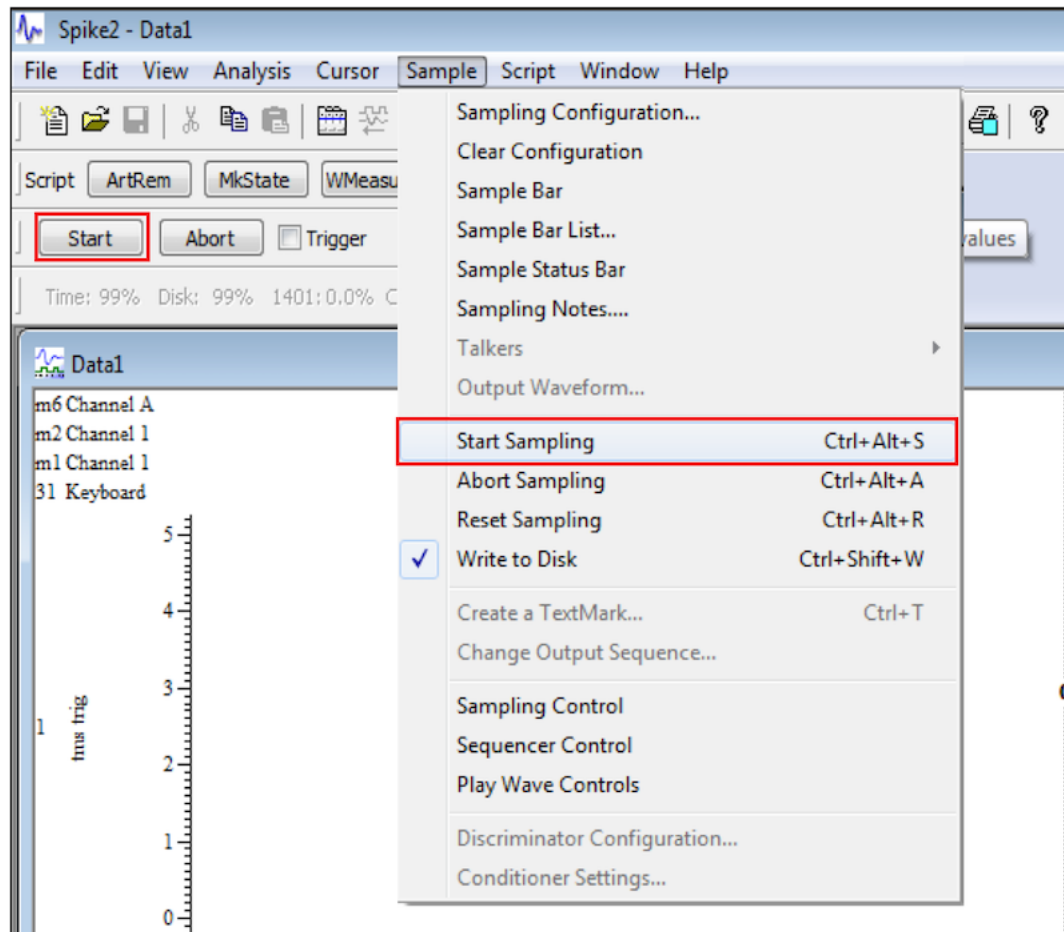
Fig. 40: The start button is equivalent to choosing Sample Start Sampling from the menu (red rectangles).
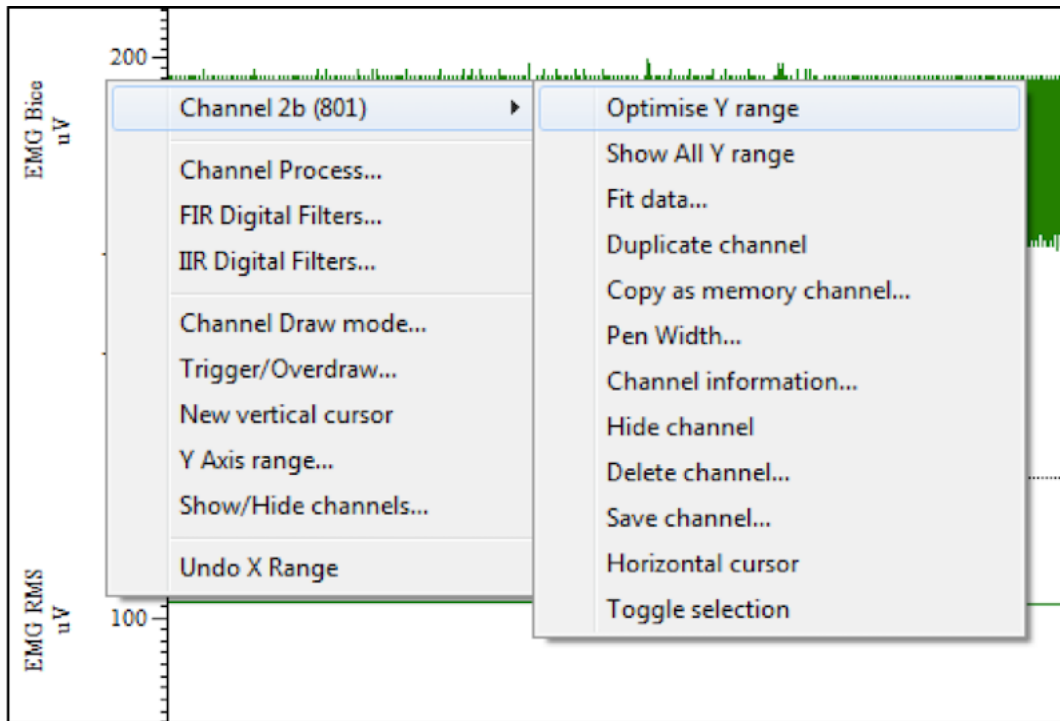
Fig. 41: Optimize the view of the readings for both yourself and the participant. Each channel in the Spike window can have its Y-Axis adjusted: either click and drag the axis or right click the Y-Axis and select **Optimize Y-Axis**.

- Occasionally someone has a higher threshold and you'll need to increase the amplitude to stimulate the hotspot (e.g., we've seen amplitudes of 75 or even 80, but this is uncommon).

- When you find a robust thumb twitch, press **Stop** on the stimulation control panel. We are done recording:

In the *Navigation control panel*, choose Instr. Marker Navigation to navigate using the instrument marker we just created.

9.17.3 Pest

Now that we have identified the motor hotspot, start Pest. Pest is a motor threshold assessment tool. Pest runs some clever algorithms to help choose the optimal stimulation threshold. We will use Pest to stimulate the identified motor hotspot from the previous step.

9.17.4 Test Threshold for 3/6

After identifying Pest's guess at the motor threshold, ensure that 3/6 stimulations at that intensity produce a twitch.

- If we get too few responses, increase the threshold by `1` and try again.

- If we get too many responses, decrease the threshold by `1` and try again.
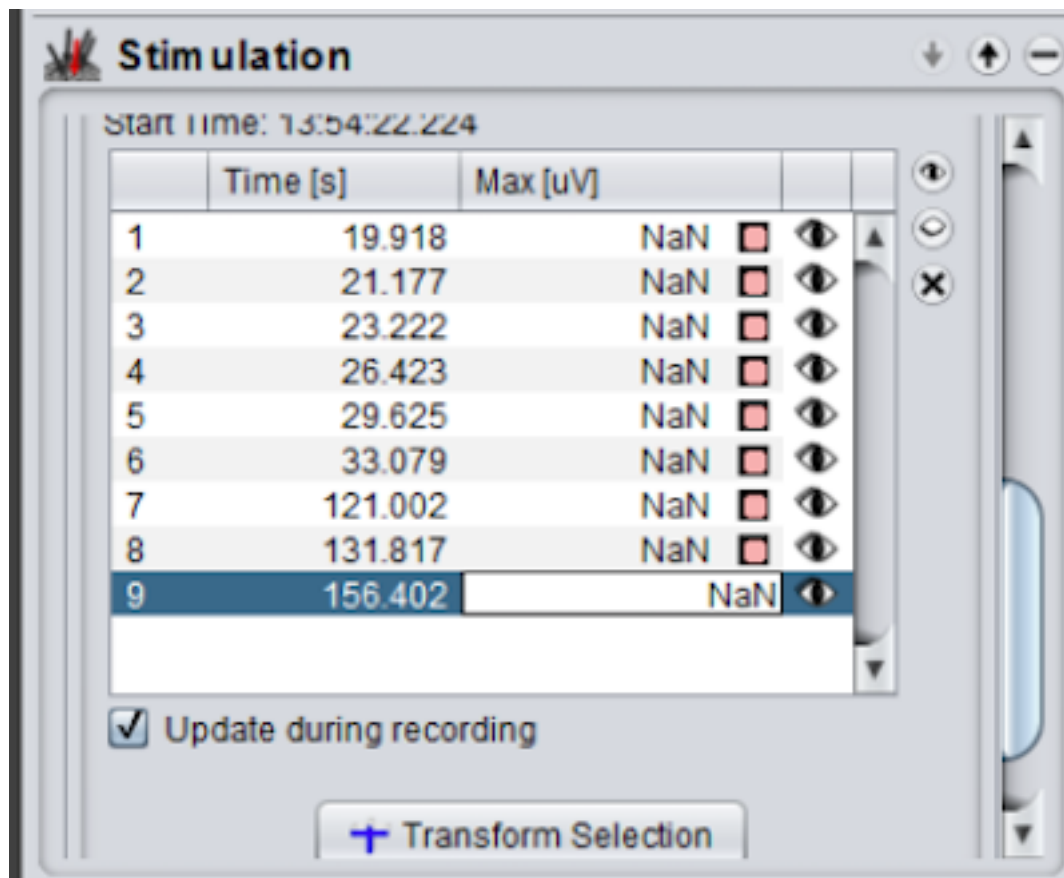
Fig. 42: Select a stimulation from the list that corresponds to a robust thumb movement. Choose **Transform Selection** to open the Instrument markers Control panel and transform your selection into an instrument marker.
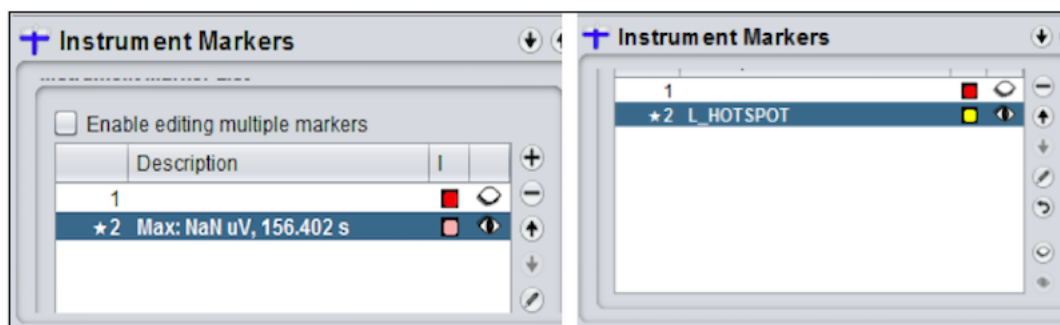

Fig. 43: **Left**: The Instrument Markers Control Panel opens and displays the stimulation you chose above. **Right**: Give the selected marker a better name and color.
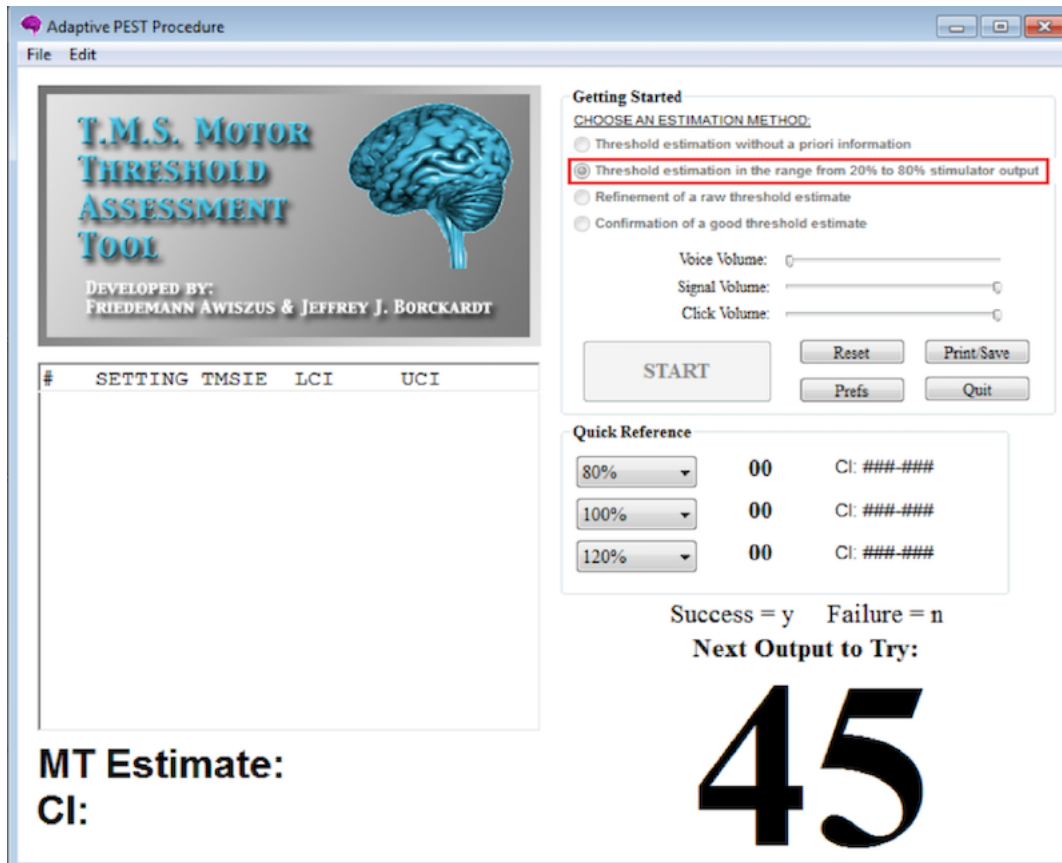
Fig. 44: Select **Threshold estimation in the range from 20% to 80% stimulator output** (red rectangle) to use the default value of `45`. Alternatively, it is possible to set a different threshold by selecting **Refinement of raw threshold estimate**, but the default is usually fine. Press **START**. `45` (or whatever threshold you chose) appears in the lower right as shown here. Pest will tell you what coil amplitude to try next. That amplitude can be set, for example, on the *stimulator panel* on the Localite machine.
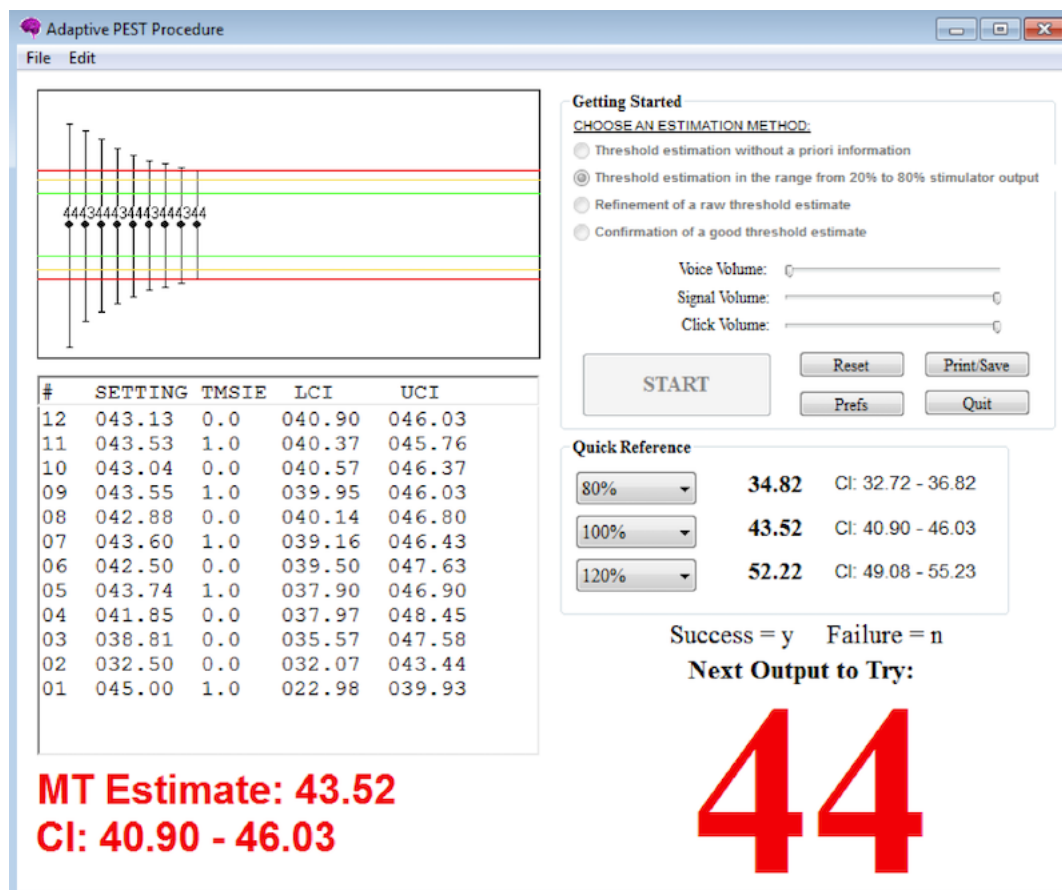
Fig. 45: Each stimulation applied to the hand knob will either produce a finger twitch or not. If the stimulation produces a twitch, then type `y` into Pest. If the stimulation does NOT produce a twitch, then type `n` into Pest. Pest will make a sound if it detects your `y` or `n` (but it can lag a little, so make sure you wait long enough to hear the acknowleding sound). Pest will suggest the next amplitude to try. Pest displays a record of the stimulations graphically and as a table on the left. Avoid clicking the table as the software will become unresponsive (this appears to be a bug in the Windows version we are using). When Pest has sufficient data, the big number (e.g., `44` in this figure) on the lower right will turn red. This is Pest's best guess at the **motor threshold**.

- If you get 4/6 and then decrease and get 2/6, then your final value is the 4/6 value.
- After we establish the RMT we calculate 70% of that value. 70% of RMT will be our TBS (Theta Burst Stimulation) intensity for both CTBS (continuous Theta Burst Stimulation) and ITBS (intermittent Theta Burst Stimulation).

9.17.5 Spike2: Quit and Stop (Do NOT close)
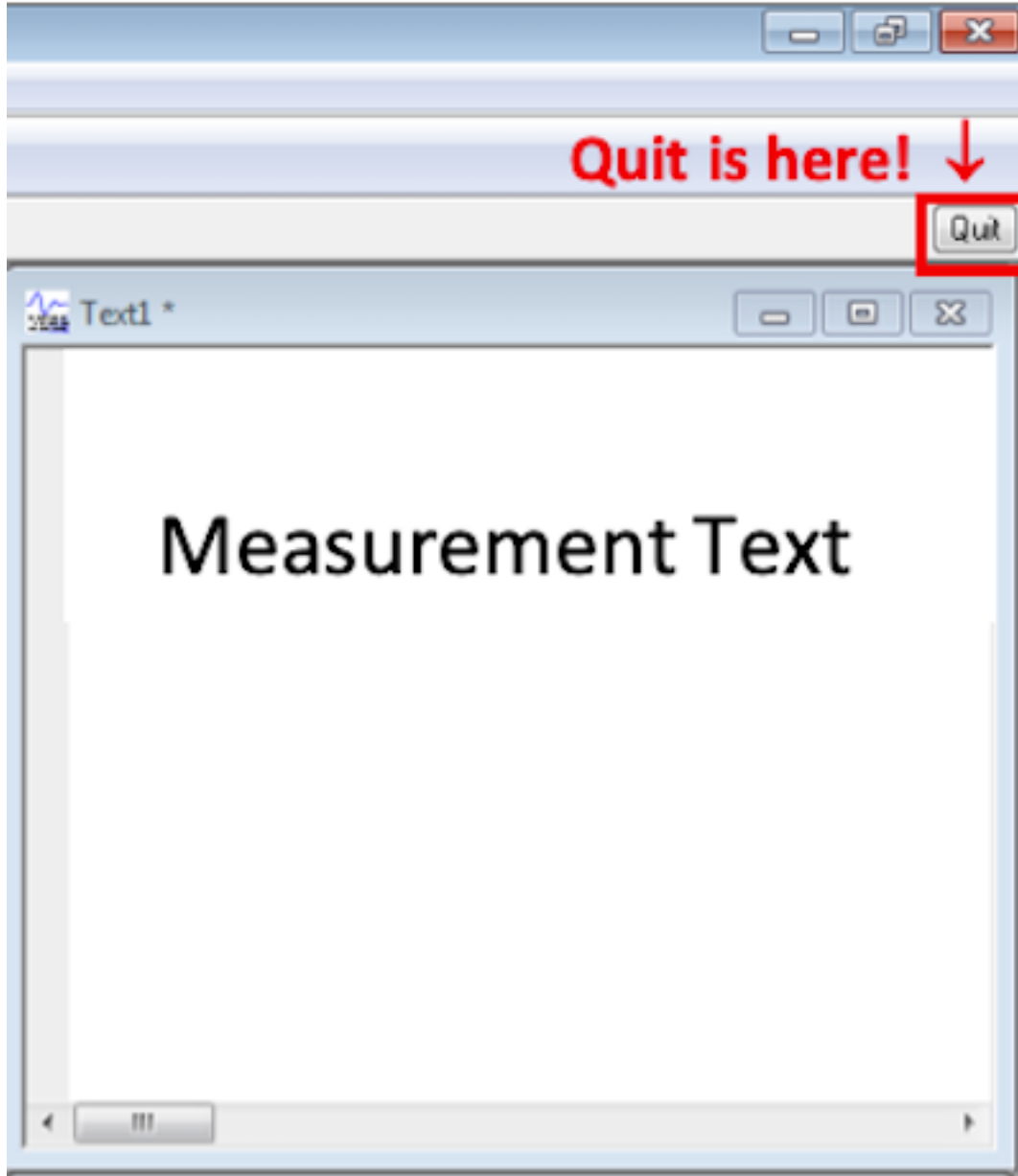
Note: When recording is stopped, it cannot be resumed. We will wait until later to save the MEP data.

- Remove electrodes from participant

- For now press the quit button (illustrated in the figure) and the stop button on the Toolbar.

- Later, when we finish the session we will save data from Spike.

- For now, you **must not close Spike**.

9.18 TBS (Theta Burst Stimulation)

- It is time to change the coil from the small C-B60 to the large Cool-B65 we will use for TBS. As soon as you unplug the small coil, the Localite computer will prompt you to *switch or keep calibration*. It does not matter what you select, or if you ignore the prompt until the large coil is plugged in.

- Ensure the cooling unit is on.

- You are going to apply either ɪTBS which increases cortical excitability or ᴄTBS which decreases cortical excitability.

- Check the positioning of the coil tracker balls to ensure they won't be in the way while you are applying the stimulation to the participant. You can move them but you should do so before you calibrate the coil.

- *Calibrate the coil*. This time choose **Calibrate Coil 2**. The calibration screen should recognize that the coil is now the Cool-B65.

- Apply the *coil setup instructions* to Coil 2 (the Cool-B65)

- On the MagPro screen press the **Main** button.

- **Main** will be replaced by the **Timing** on the bottom left of the screen.

- Turn the knob to ʏ for iTBS or ᴄ for cTBS. Press **Recall**.

- Press **Timing** to see the timing details. The START/STOP softkey on the lower right is enabled. Now the pulse train can be triggered from the **Stimulator** panel using the **Start/Stop Train button**

- Navigate to the target of stimulation (something other than the hand knob, we presume).

The Cool-B65 coil is heavy which makes it difficult to hold steady for the requisite amount of time. The heavy coil may also tend to move the participant's head as it rests against it. You may find it helpful to have the participant use the chinrest (there is a soft blue squishy disk to make it more tolerable).

9.19 Wrap Up the Session

- You want to save data from both the Localite and MEP machines to your external device.
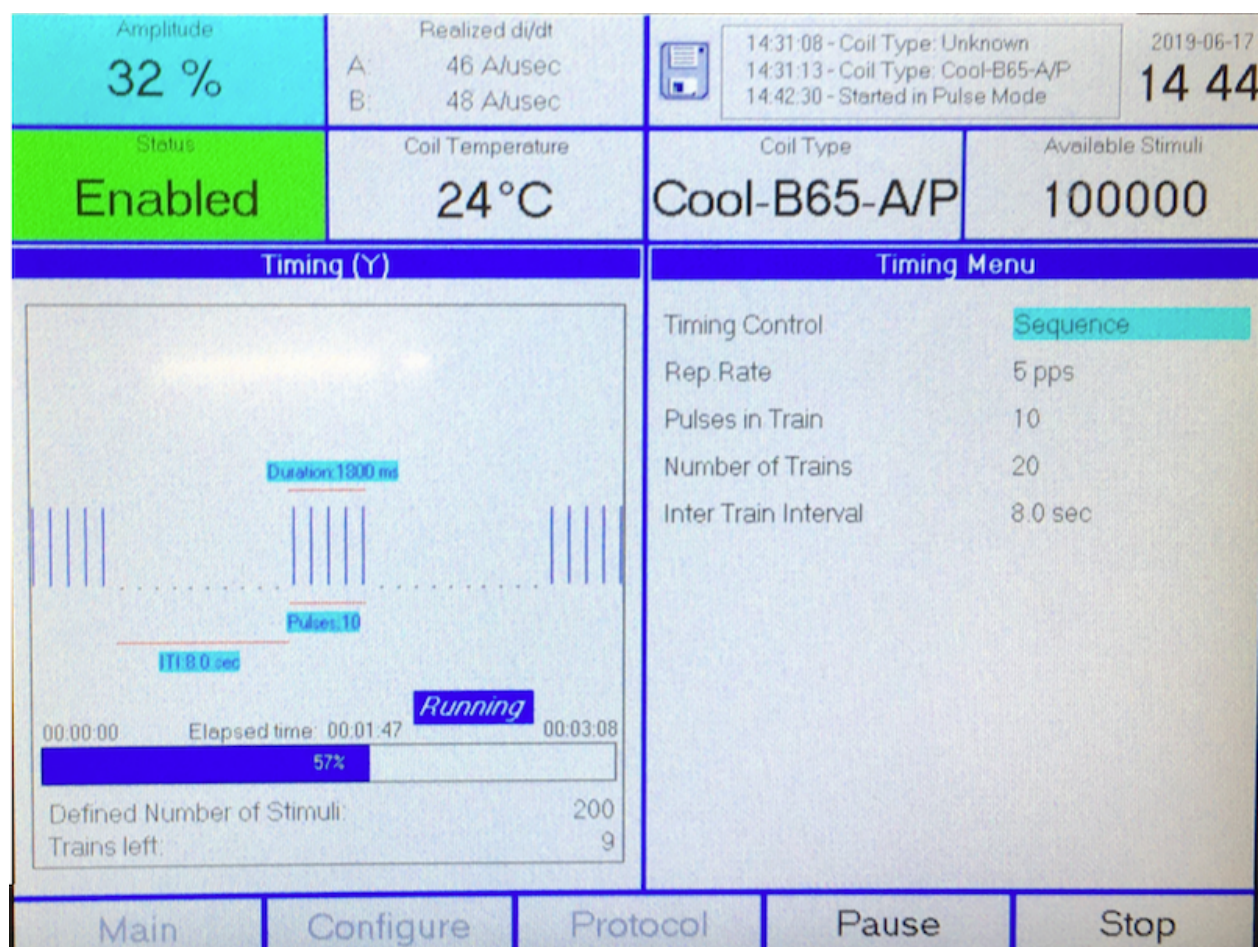
- You also want to turn off hardware.

Fig. 46: The MagPro screen with 32% amplitude, and the Cool-B65 coil enabled: The status area now reports the coil temperature and available stimuli. The Timing screen is displayed, instead of the Main screen. This is the timing for Y, the iTBS sequence. Note the long (8 sec) intertrain interval and the STOP softkey on the lower right.
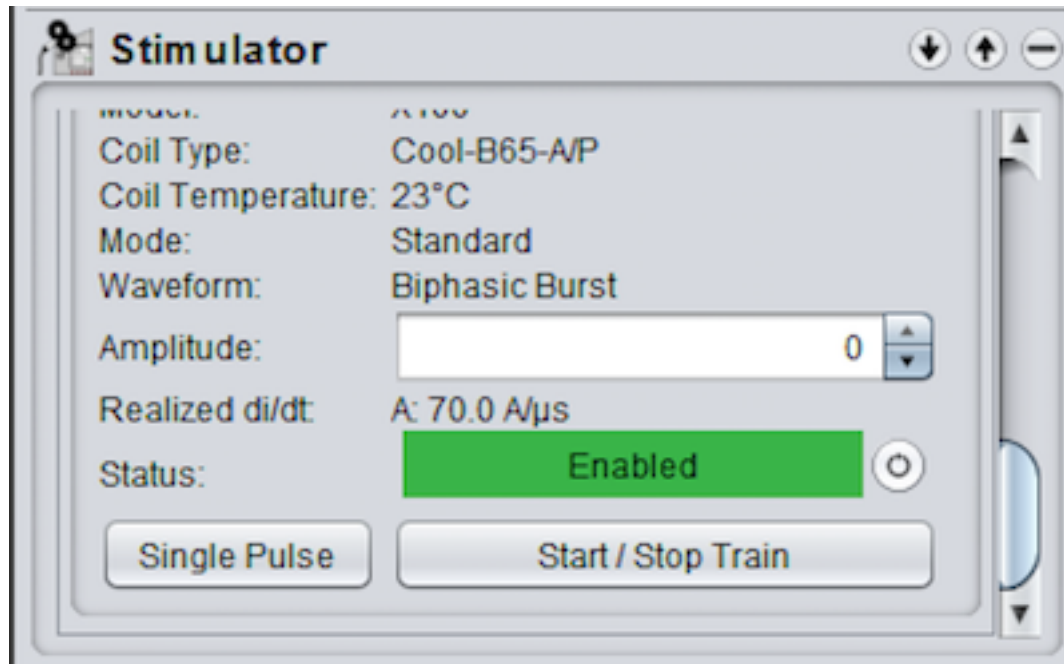
Fig. 47: The stimulator panel with the Cool-B65 enabled: Because the Timing screen is displayed on the MagPro, the **Start/Stop Train** button can be used to initiate the pulse train. Of course, before initiating the pulse train, the amplitude should be set to the value you calculated earlier as 70% of the resting motor threshold.

9.19.1 Save Data

Localite Data Save the participant folder you created earlier to your external device. **MEP Data** Save data from Spike2 and Pest to your external device.

- **Spike2**

    - **Save Main EMG Reading Window**: Make sure that the **focus is on the main EMG Reading Window** and click *Save* in the upper left to save it to the appropriate participant folder. If the file has the *.smr extension, then you know you saved the right window.

    - **Save Log**: Save the log by clicking in the log window and following the same steps (but the output will be a *.txt file).

    - Ensure that you have named both files using your protocol and that they are in the participant folder you created.

    - Copy the participant folder to your external drive.

    - You can close Spike2.

- **Pest**

    - Save the Pest data to the participant folder.

9.19.2 Turn off Hardware

- **MEP** Turn off the amplifier and data acquisition devices.

Fig. 48: **Left**: In its vertical poisition the chinrest is a good height for the average participant. It can easily be extended to be taller. If the chinrest is too tall for your participant, you can tip its angle down to any height (using the adjustment marked with the red arrow). **Right**: The chinrest tipped at an angle.

- **Coil** Disconnect the large coil, and reconnect the smaller coil.

- **MagPro** Turn off the Cooling Unit and MagPro.

- **Localite Computer** Turn off Localite computer.

Note: Order is not terribly important here. The MEP equipment is separate from the Coil/Localite/MagPro. It is probably best to turn off the Localite machine last (but I'm not sure why I say that).

FSL

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu

Date Created: 2018_06_14

Date Updated: 2018_11_14

Tags: Neuroimaging software, DWI

OS: UNIX (e.g., Mac or Linux)

FSL is a general purpose neuroimaging workbench, like AFNI and SPM. FSL excels at handling diffusion weighted imaging.

FSL has special facilities for using Siemens *field maps* for distortion correction of epi images (both dwi and fMRI) using the script epi_reg to run boundary based registration (*BBR*). Field maps can be acquired quickly at scan time (about a minute). Using field maps for distortion correction requires correctly identifying some image parameters that are described on the *Glossary* page: *Difference in Echo times*, *Dwell time*, and *Phase Encode Direction*.

In addition, *reverse phase encode* images can be used to enhance distortion correction. Mostly these corrections are used for dwi, where the distortions are especially bad, but some people have successfully used reverse phase encode distortion correction on fMRI. A couple of reverse phase encode B0 volumes can be acquired quickly at scan time (about a minute). Using the reverse phase encode volumes for distortion correction requires the use of *topup*, and optionally *applytopup*. It is also necessary to correctly identify some image parameters that are described on the *Glossary* page: *Phase Encode Direction* and *Total Readout Time*.

ITK-SNAP

Maintainer: Dianne Patterson, Ph.D. dkp @ email.arizona.edu

Date Created: 2018_10_24

Date Updated: 2019_07_30

Tags: lesion, segmentation, anat

Software: at least ITK-SNAP 3.8 beta (though 3.6 is generally okay)

OS: Windows, Mac or Linux

11.1 Introduction

ITK-SNAP is an image segmentation tool. ITK-SNAP is similar to MRIcron in that both are excellent tools for creating lesion masks on MRI images. ITK-SNAP may have some advantages: smarter fill tools (*adaptive paintbrush* and *automatic segmentation*) in addition to *3D editing tools*. Because MRIcron assumes isotropic voxels (or near isotropic), it is not well suited to drawing on clinical data which often has high resolution in-plane but very thick slices. However, it is worth noting the ITK-SNAP needs isotropic (or near isotropic) data to do good semi-automated segmentation. Here is a simple FSL-based script to create isotropic images: iso.sh.
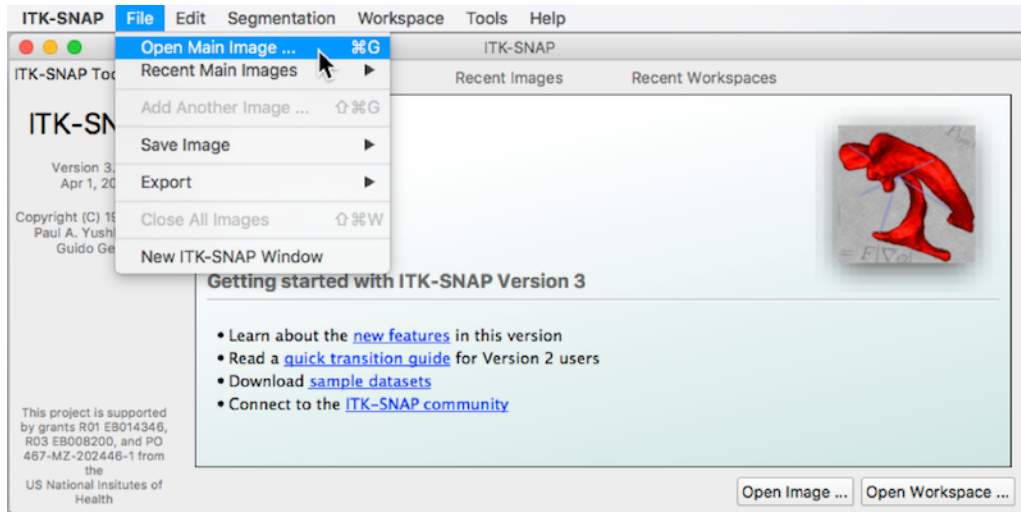
ITK-SNAP Documentation

ITK-SNAP Video Library

See the *ITK-SNAP Segmentation of Stroke-related Lesions page* for a link to some example data and instructions for using automatic segmentation for filling the lesion.

11.2 Load and View the Image / Images

- Locate the ITK-SNAP icon and double-click it.

- Select *File Open Main Image* from the menu.

- An *Open Image* window appears (See figure below).

- Press Browse to search for the MRI image Select the image *T1w_defaced.nii.gz* and press *Open*.
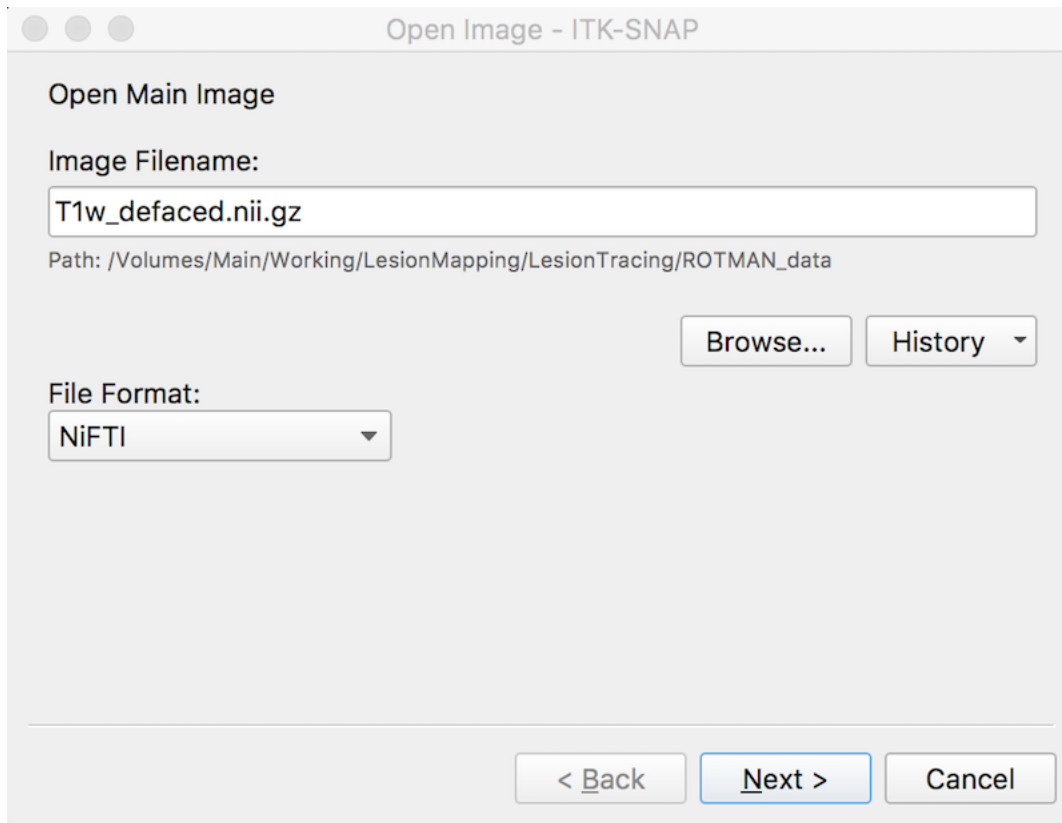
- Press *Next*. You see summary information and a warning about lack of precision. This is fine.

- Press *Finish*. ITK-SNAP's main window should now display three orthogonal views of the MRI scan.

- If the images are too dark, adjust the contrast: *Tools Image Contrast Auto Adjust Contrast (all layers)*.

- Press the *zoom to fit* button (red box, bottom right, in the figure below) so the head fills the available space.

The dark area (enclosed in the green box in the above figure, axial and coronal views) is lesion. You should be able to distinguish both the CSF-filled region, and the area of damaged tissue around the rim. Some ventricle is unavoidably also included in the box.

If you have multiple images that would be useful for drawing the lesion, you can choose File Add Another Image. Snap will display all the images you have and yoke them together, even if they are different resolutions.

11.3 Understanding Labels

Segmentations (a.k.a. masks, labels) are simple images containing a single value. By default, most masks have the value 1 inside and 0 outside. However, if we want a mask that covers lesion and a different mask that covers ventricle, then we need masks with different values. For example, 1 could correspond to lesion and 2 could correspond to ventricle. Colors and names can be assigned to each label. ITK-SNAP has 6 default labels.

11.3.1 Label editor

You can access the label editor by clicking on the palette icon in the *Main Toolbar* (green box, figure below).

The label editor looks like this:
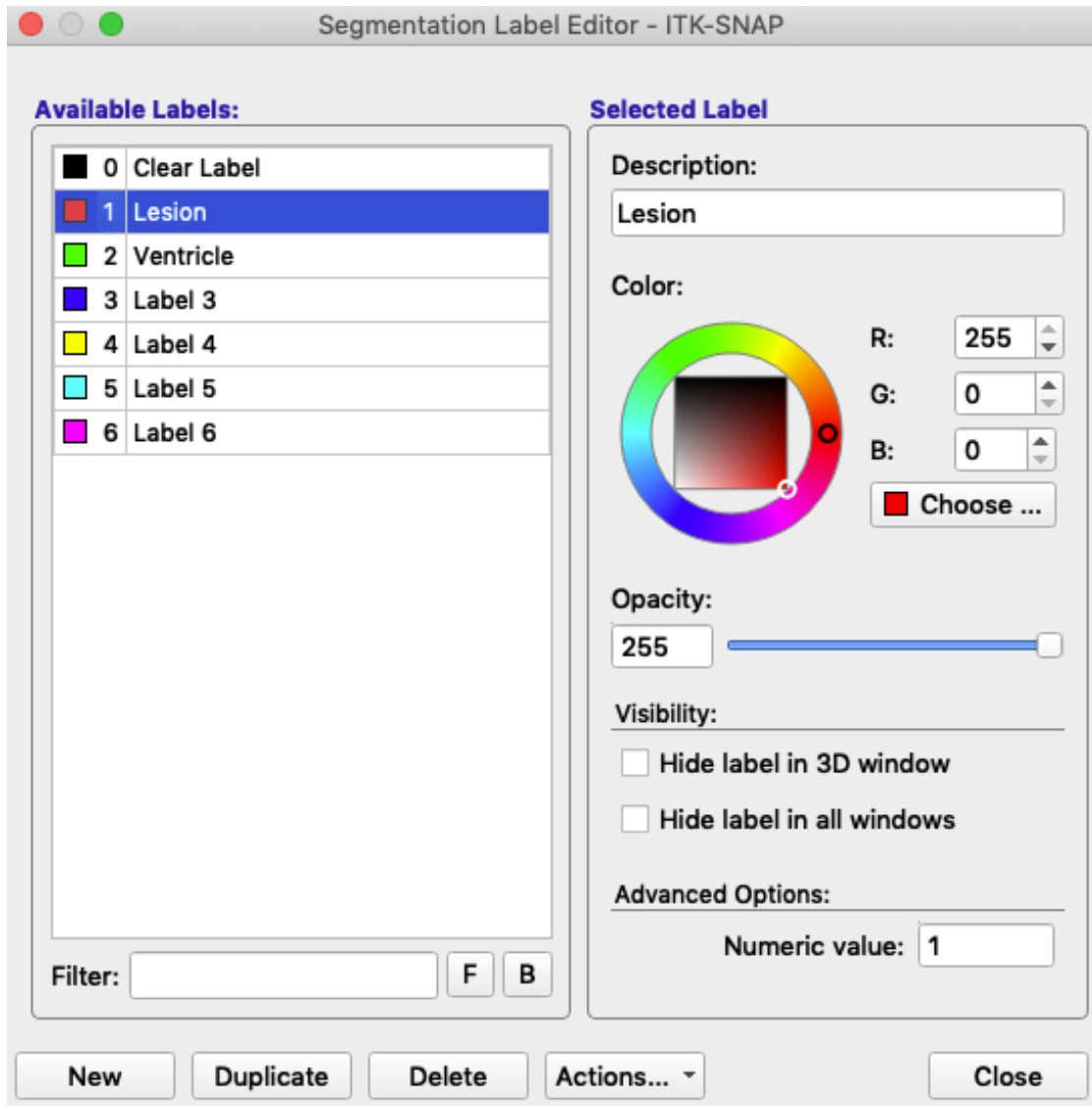
The label is defined by its *Numeric value* (lower right). Description and associated color (right, figure above) can be modified. In this figure, the label with the value *1* remains red, but has been renamed *Lesion*. The label with the value *2* remains green, but has been renamed *Ventricle*. At the bottom of the label editor, click *Actions* to see options to import or export a set of label descriptions.

11.3.2 Segmentation labels

On the bottom left of the ITK-SNAP interface is the *Segmentation Labels* section. As shown in the animated gif below, you can select the active label (i.e. the label value that will be applied by any of the 2D editing tools or automatic segmentation) AND you can select *Paint over*.

- If you select *Paint over All labels*, then the active label will replace any value it finds.
- If you select *Paint over All visible labels*, then the active label will replace any other segmentation label (but not the clear label).
- If you select *Paint over Clear label*, then the active label will apply only to areas of the image where there are no visible labels.
- If you select *Paint over Label 4*, then the active label will replace any areas that currently contain label 4, but no other areas. For example, if you have a brain mask with the label 4, and you paint over label 4, then you will never accidentally paint over areas outside the brain, because areas outside the brain will have the clear label.
- You can reduce the opacity of a label (like the brain mask) without affecting its functionality (i.e. even if the brain mask is completely transparent, it still behaves the same way).

> **Warning:** Be careful about the *Active label* and *Paint over* settings! The feature is powerful.
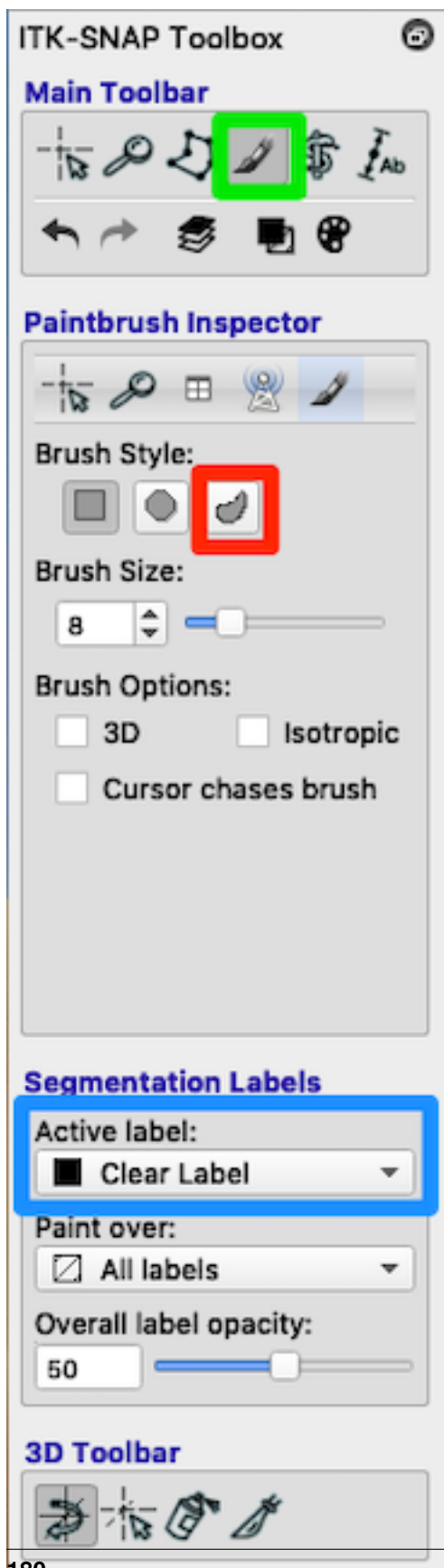
11.4 2D Segmentation Editing

- If you have created a workspace, for example by following the instructions here: *ITK-SNAP Segmentation of Stroke-related Lesions page*, load that workspace now (e.g. Workspace → Open Workspace).
- Any changes you make to the segmentation can be saved with the same name. Be cautious about changing the names.
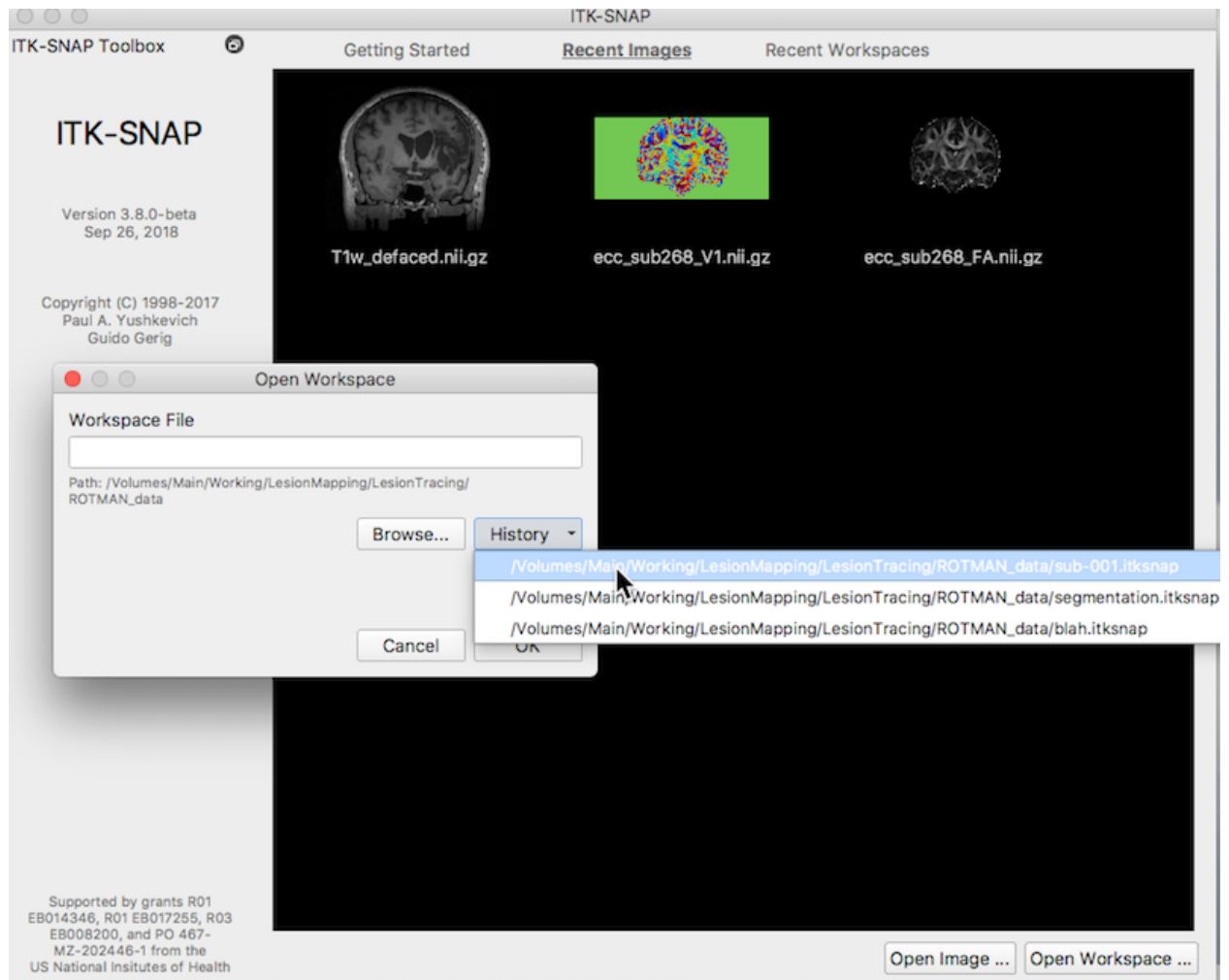
11.4.1 Paintbrush

- Look at the toolbox. Choose the paintbrush (green box in above figure).

- **Paint** If you need to manually paint additional areas (e.g., parts of the lesion not filled by the algorithm), then set *Active label = Lesion*.

- **Paint and Erase Mouse buttons**: The left mouse button paints and the right mouse button erases.

- **Erase with Clear Label** If some of the lesion mask leaked into the ventricles you can use the paintbrush to erase it. Look at Segmentation Labels (blue box in figure). Painting with the *Clear Label* and *Paint over: Lesion* will erase lesion that has leaked into the ventricles.

11.4.2 Adaptive Brush

See the red square in the *Paintbrush Inspector* in the above figure. Try a relatively large brush size (e.g., 16 instead of 8), and try the 3D brush. Both of these choices should reduce the number of clicks you need to fill a region. The adaptive brush will paint everything that has an intensity similar to the intensity at the crosshairs AND is contiguous to the crosshairs. That is, if you have a large adaptive brush that straddles a tissue boundary, the fill will apply only to voxels contiguous to the crosshairs. The adaptive brush can be useful for filling in ventricles, which have a fairly consistent intensity, but very curvy borders. Click each time you select a new position for the crosshairs.

Note: Imagine that you have labeled the ventricle and you want to paint lesion right next to it without overwriting any of the ventricle. To preserve your ventricle, but paint lesion very close to it, choose *Paint over = Clear Label* instead of *Paint over = All labels*. The *Clear Label* is the default that covers all areas of the image where there is no visible label. *Paint over = Clear Label* will prevent the brush from overwriting another visible label.

Note: *Overall label opacity* (bottom of the figure) applies only to **viewing** the lesion segmentation label. It is nice to have this be translucent so you can see where to edit.


- The **Adaptive Brush** has two settings: *Granularity* and *Smoothness*. It is not immediately obvious what these mean, so below are some examples of creating a single brush stroke, size 20, at the same coordinates with different *Granularity* and *Smoothness* settings. Look at the yellow blob at the crosshair location:
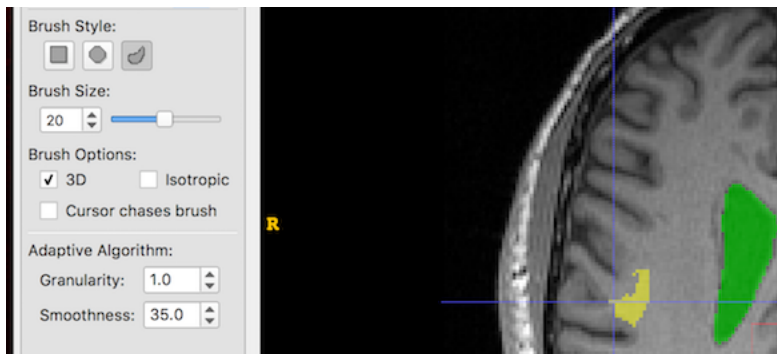
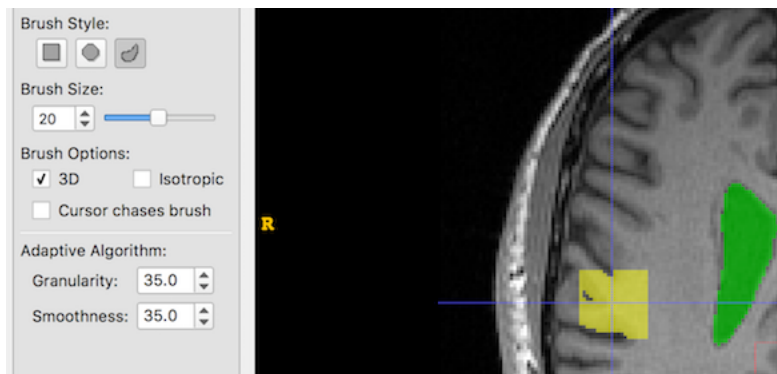First, we set *Granularity* and *Smoothness* to their minimums:

And now we increase *Granularity* only. Low granularity values lead to oversegmentation, while higher values lead to undersegmentation:
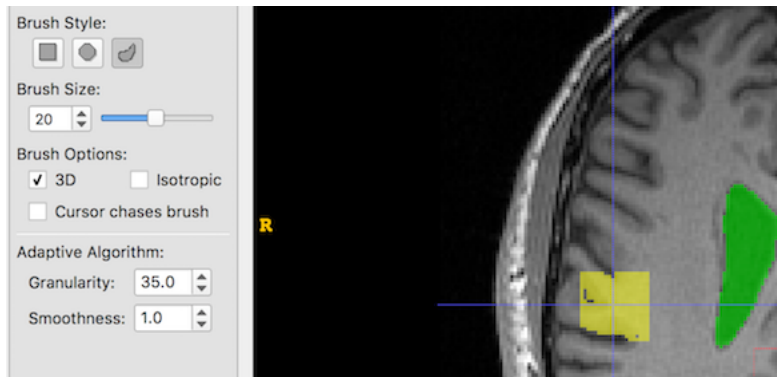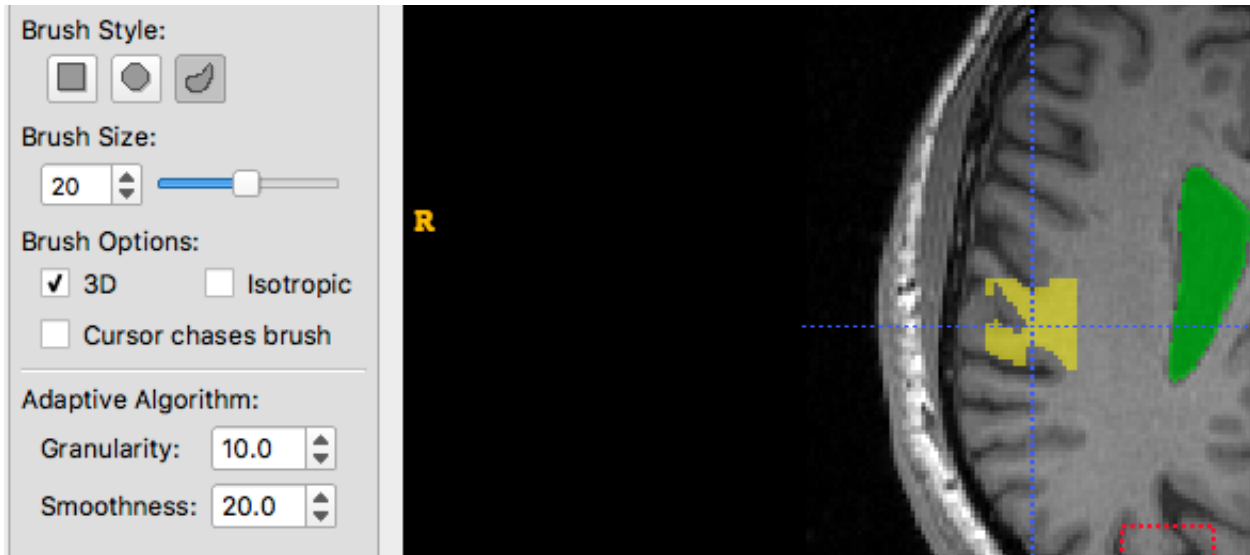
And now we increase *Smoothness* only. Larger smoothness values produce smoother segments:

And now *Granularity* and *Smoothness* are both increased:

Finally, we try to find an appropriate balance:

11.4.3 Polygon

- Immediately to the left of the paintbrush on the Main Toolbar, is the polygon tool (green box in figure below).

- Select it and you'll see the polygon inspector.

- Now you can draw around the structure of interest insuring that you are using the correct active label. To draw a polygon click the mouse to start, and then click again to define each line segment. You will need to close the figure.
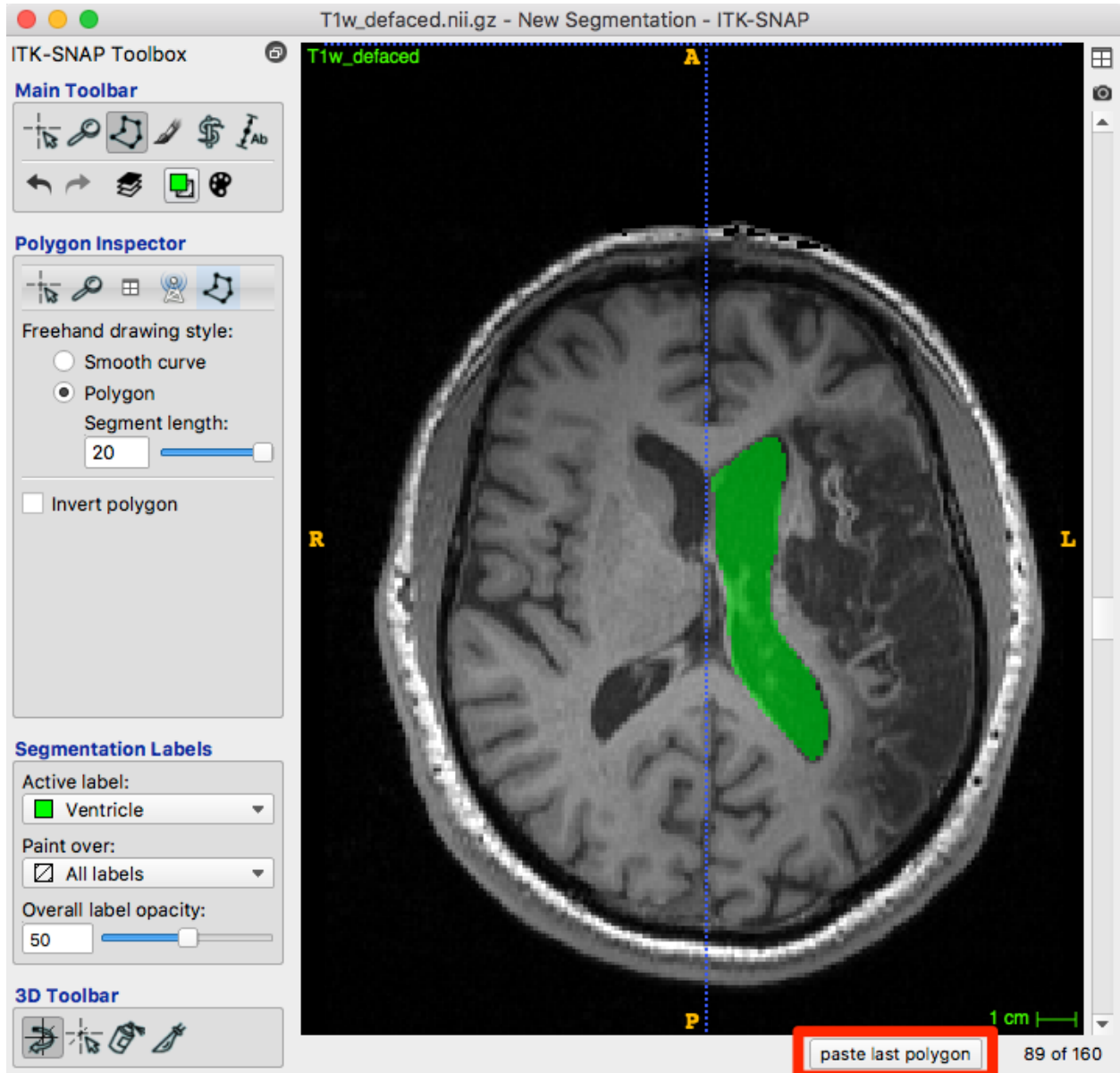

Note: Although we use the ventricle label in the figure below, the polygon is drawn in the default red.

When you are happy with the polygon, click *accept* (red box, figure below).
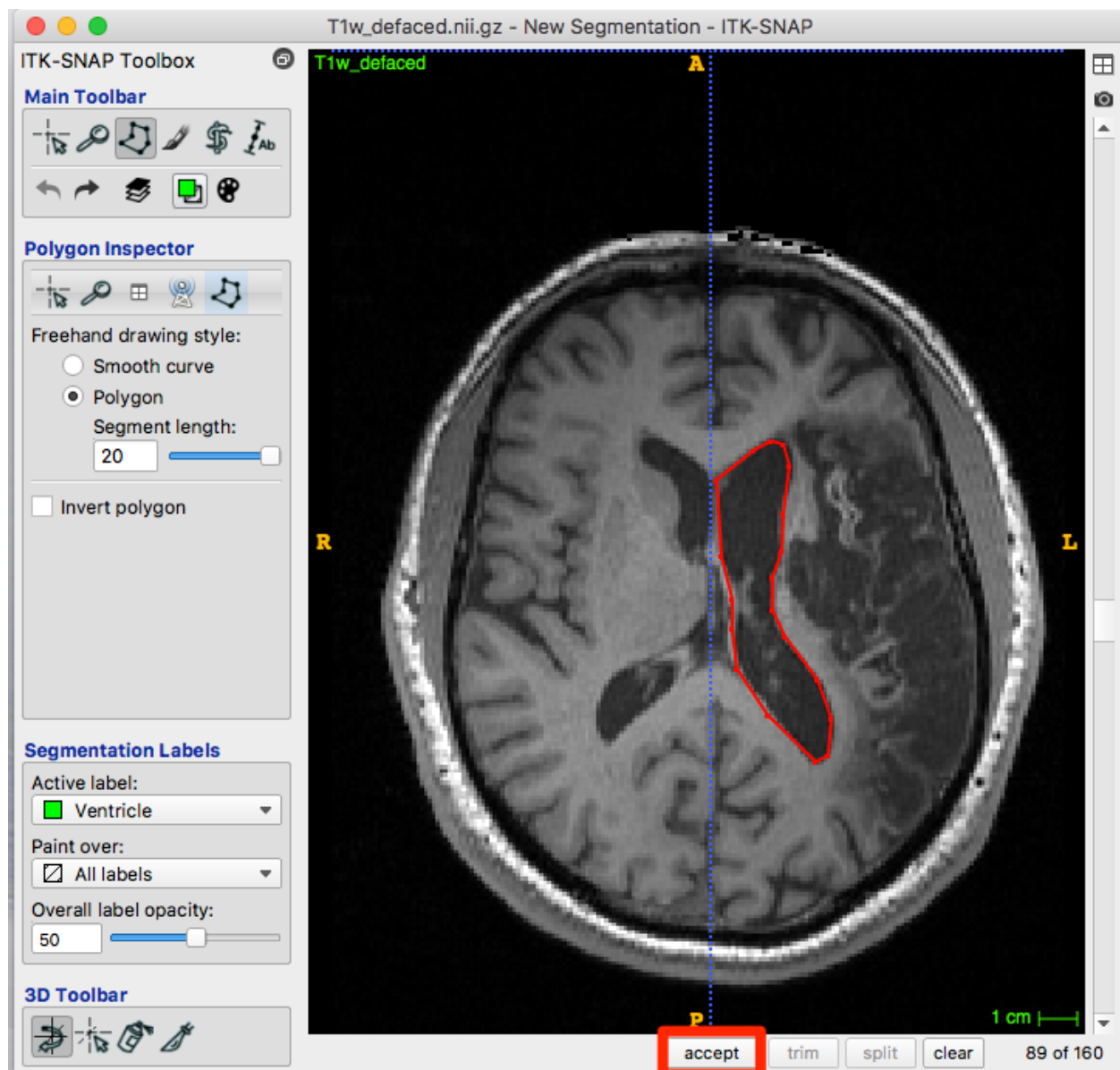
After choosing *accept*, the roi is created with the appropriate color for your label:

Below the image is a *paste last polygon* button (red box, figure below). Move to the next slice and click the button. A new polygon appears...you can drag vertices or edges, create new vertices or trim them until satisfied. Accept and repeat.

11.5 3D Segmentation Editing

3D editing tools only work once you have created a segmentation.

11.5.1 Scalpel

- You can create a plane through the 3D image and relabel values on the other side as something different.

- Set an active label other than the clear label.

- Identify a plane that roughly separates ventricle from lesion.

- Select the scalpel tool at the bottom of the toolbox (green box in figure below).

- You can manipulate the position of the cut plane with the arrow point, central ball or tail.

- When you are satisfied with the cut plane, click *accept* and *update* (blue box, bottom left).
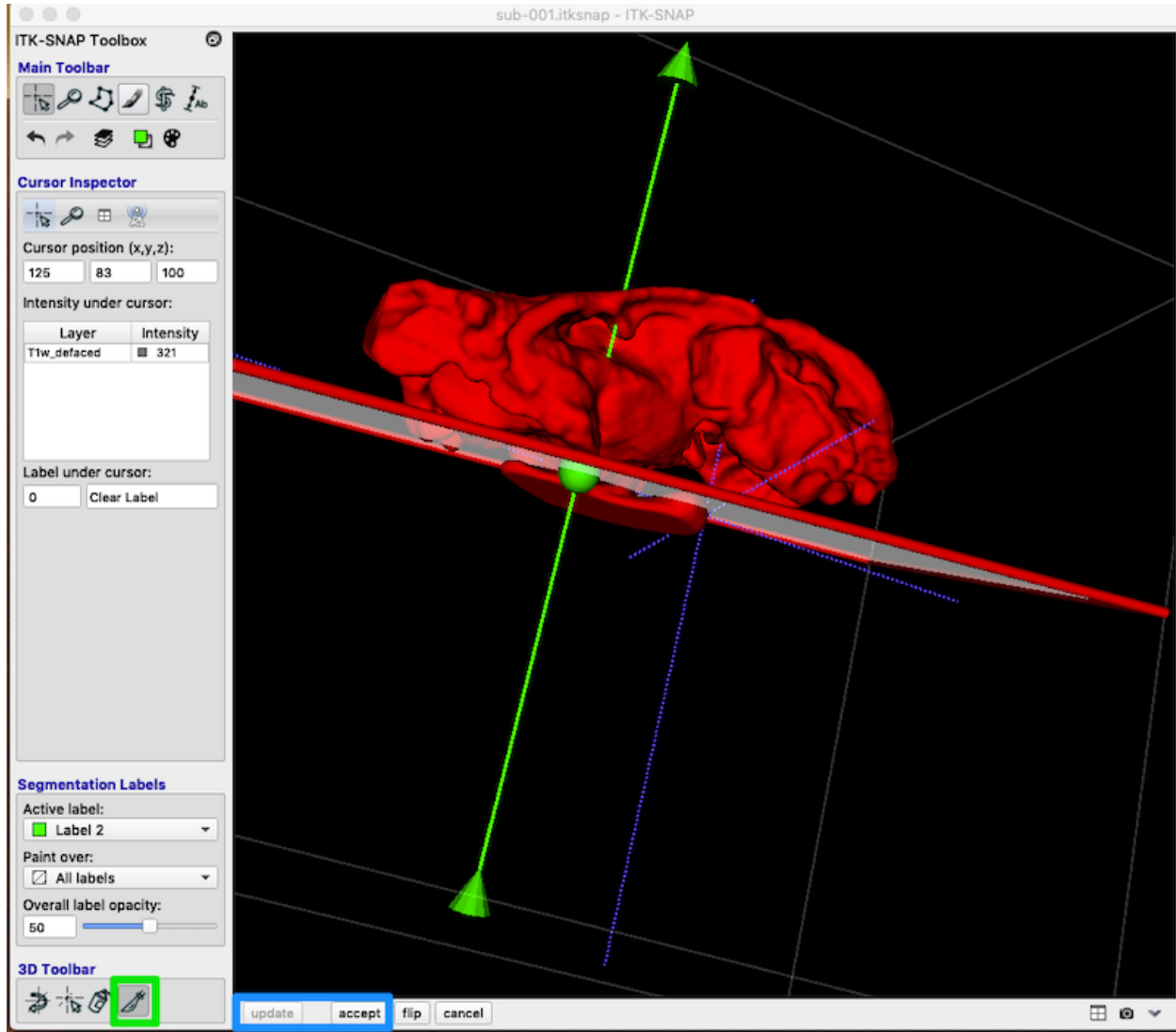
Your revision should look something like this (I have changed *Label 2* to *Ventricle*):
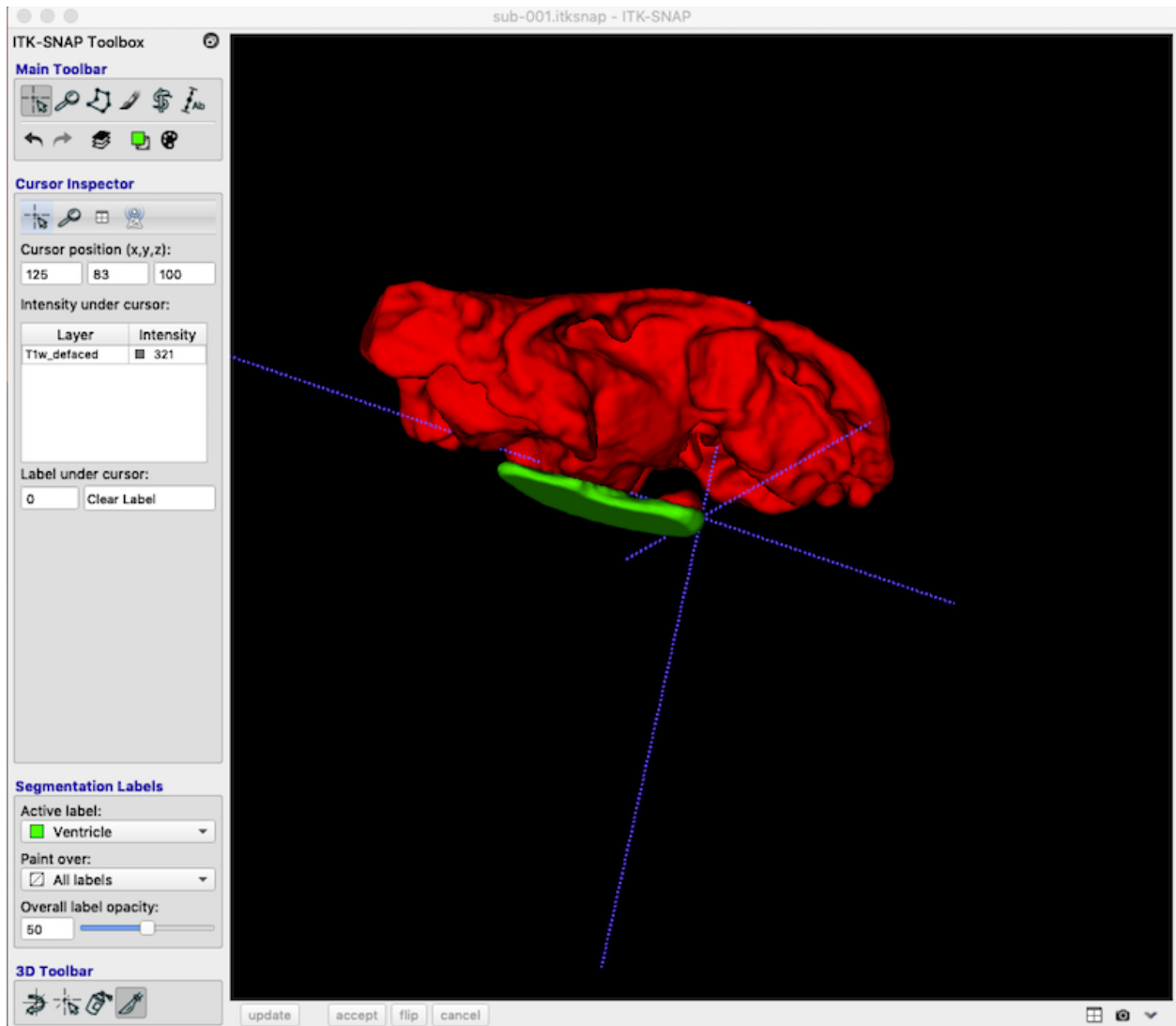
11.6 Add a Second Segmentation

As you can see, you are not limited to a single segmentation type. Following the procedures above I have grown bubbles in the ventricle and added these to the final segmentation. Lesion is associated with the value 1. Ventricle is associated with the value 2.
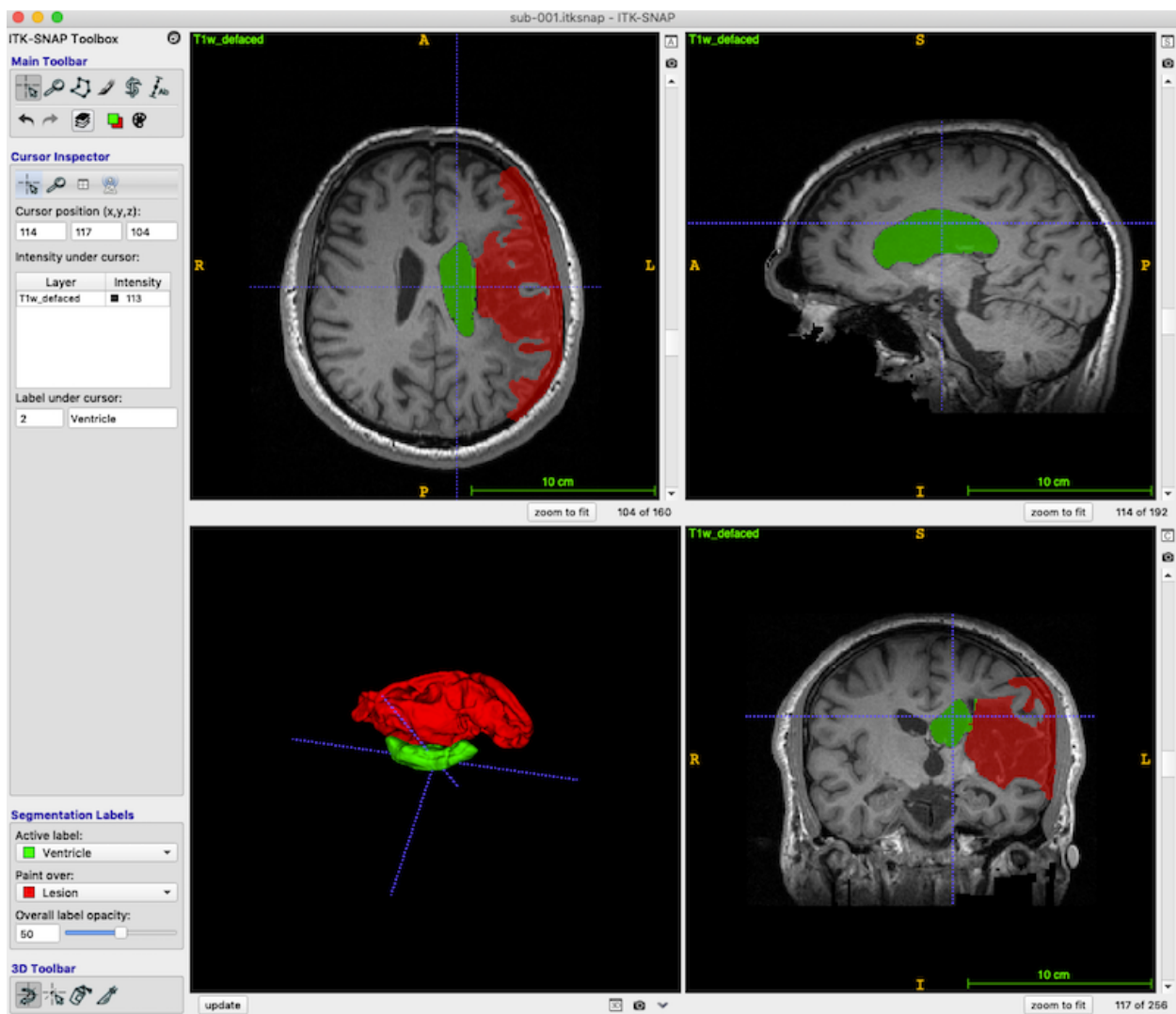
11.7 View Volume Statistics

ITK-SNAP can provide volume information for each segmentation

Mango

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2019_03_01
Date Updated: 2019_07_25
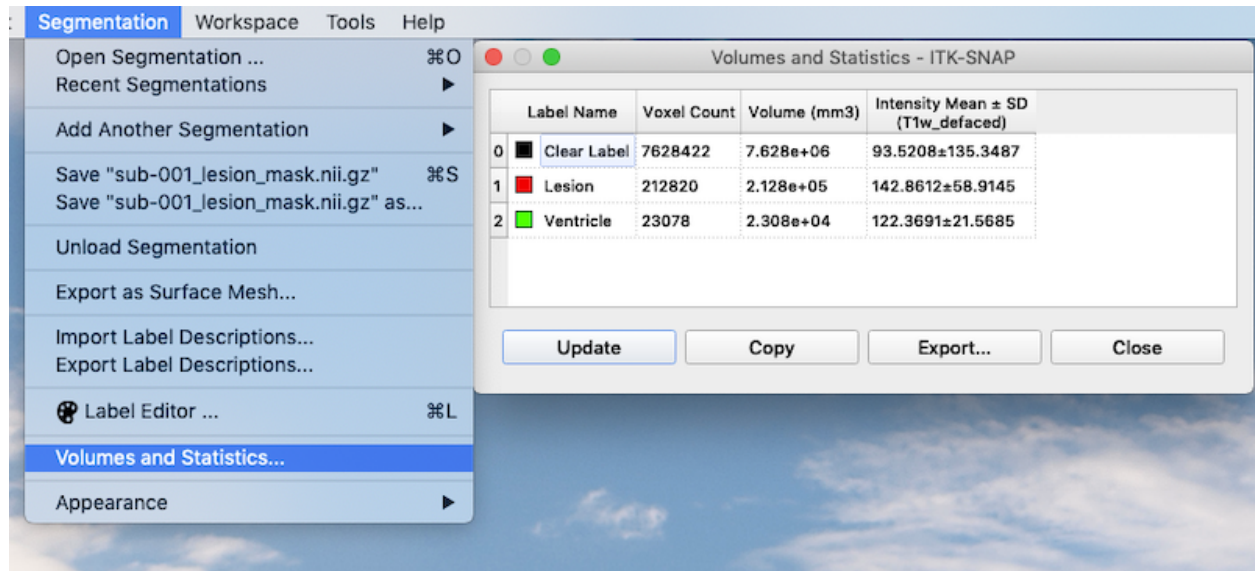Tags: Neuroimaging software, viewer
OS: UNIX (e.g., Mac or Linux) and Windows

Mango is a viewer for MRI and a tool for creating and editing masks. It is similar to MRIcron and *itk-snap*.

> **Warning:** MRIcron assumes isotropic voxels (or near isotropic). This means it is not well suited to drawing on clinical data which often has high resolution in-plane but very thick slices.

Mango has good online documentation. Below I describe several features that are especially valuable.

12.1 iMango

Mango has a companion **iPad app** called iMango which is very nice for viewing NIFTI images and editing masks. If you use the Apple pencil, or even a stylus, iMango is much nicer than drawing with a mouse. Furthermore, by working with a native iPad app, you are untethered from your computer (you can sit in front of the TV and edit for hours). There are multiple options for data transfer to and from the iPad.

12.2 Edit a NIFTI header

Mango has an **easy NIFTI header editing tool**. There are command line NIFTI header editors in the FSL Utilities (fslhd and fsledithd) and in AFNI (nifti_tool), but they are much more complicated. Why do you want to edit the header? The most common problem is that the image dimensions are wrong. This is especially true if you have a

NIFTI image that was constructed in a peculiar way (like from a stack of jpgs). Below is a description of the editing process.

- Always make a copy of your valuable image before changing the header information. It is possible to corrupt the image and never be able to open it again.

- *File Image info*
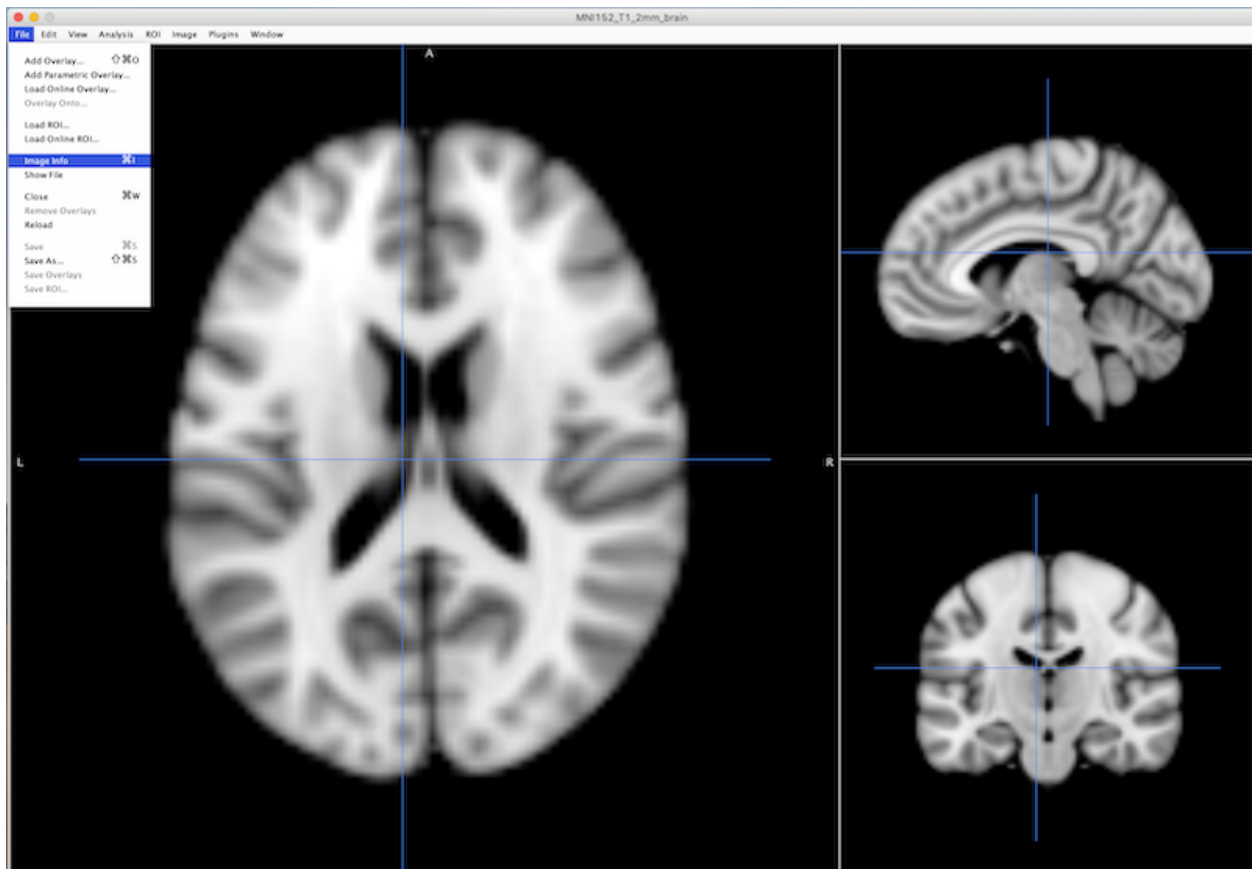
- Choose *Header* (next to the *Summary* tab at the top of the interface: black square on left, figure below).

- Choose *Edit* (orange square on bottom left of figure)

- The x,y and z dimensions are in red, green and blue boxes respectively on the right of the figure.

- Choose *Save and Reload* (bottom right of figure).
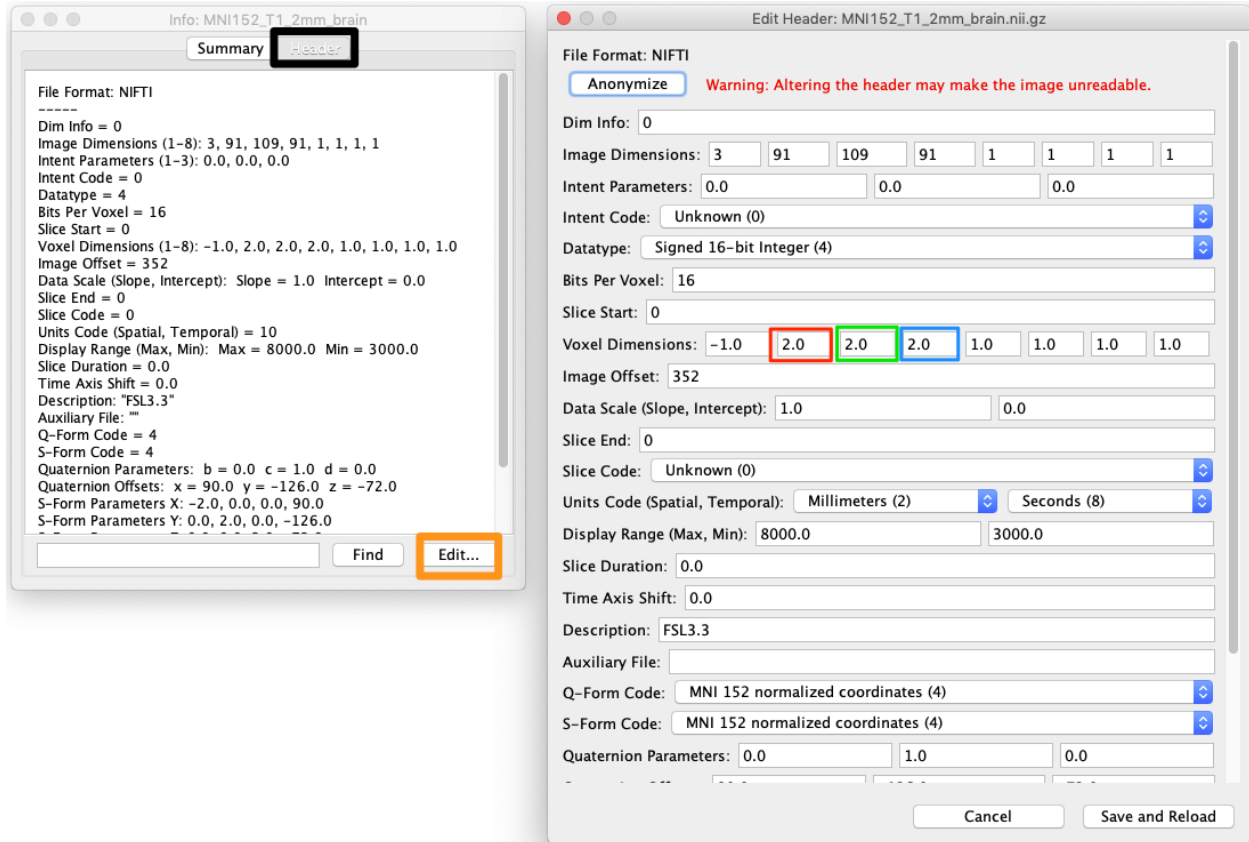
That's it.

12.3 Create a NIFTI viewer Webpage

You can use Mango to create an HTML page that displays a self-contained interactive NIFTI image viewer. Simply click the single HTML file on disk to display your images in a web browser | N.B. I have had problems with Internet Explorer, but Firefox and Chrome work well.

12.3.1 Why an HTML NIFTI Viewer is Useful

- The HTML viewer can be used by anyone, whether or not they have neuroimaging software installed.

- In Powerpoint, you can insert a relative link to any object, including the HTML NIFTI viewer. Instead of having a static image in your neuroimaging presentation, or having to open a viewer and load the relevant images, you simply click the HTML page and you can move around in 3D and control overlay opacity.

    – If you use a picture of the NIFTI viewer for your Powerpoint link, it looks nicer and can substitute if something goes wrong with the link.

    – It helps to keep the HTML page in the same folder as your Powerpoint.

    – Keep in mind, that when you click the link on Mac, Powerpoint keeps the focus, so you don't see the HTML page (you can use command-tab to bring it to the forefront).

    – On Windows, the HTML page gets the focus as soon as you click the Powerpoint link.

    – I have had problems with Powerpoint changing the links I define, so recheck these before a presentation.

- It is possible to post your HTML viewers to a Papaya web server, which you can install using a web server like Tomcat

- If you don't want to install a web server, then you can go to this web address http://rii.uthscsa.edu/mango/ papaya/index.html and drag NIFTI or DICOM files onto the interface. * You can even choose File Add DTI vector series and choose the FSL V1 image. Then click the color rainbow icon on the upper right and you can modulate the image with an FA image.

12.3.2 How to Make an HTML NIFTI viewer

- With an image, and any overlays of interest displayed in Mango, choose *Image Create Webpage*.
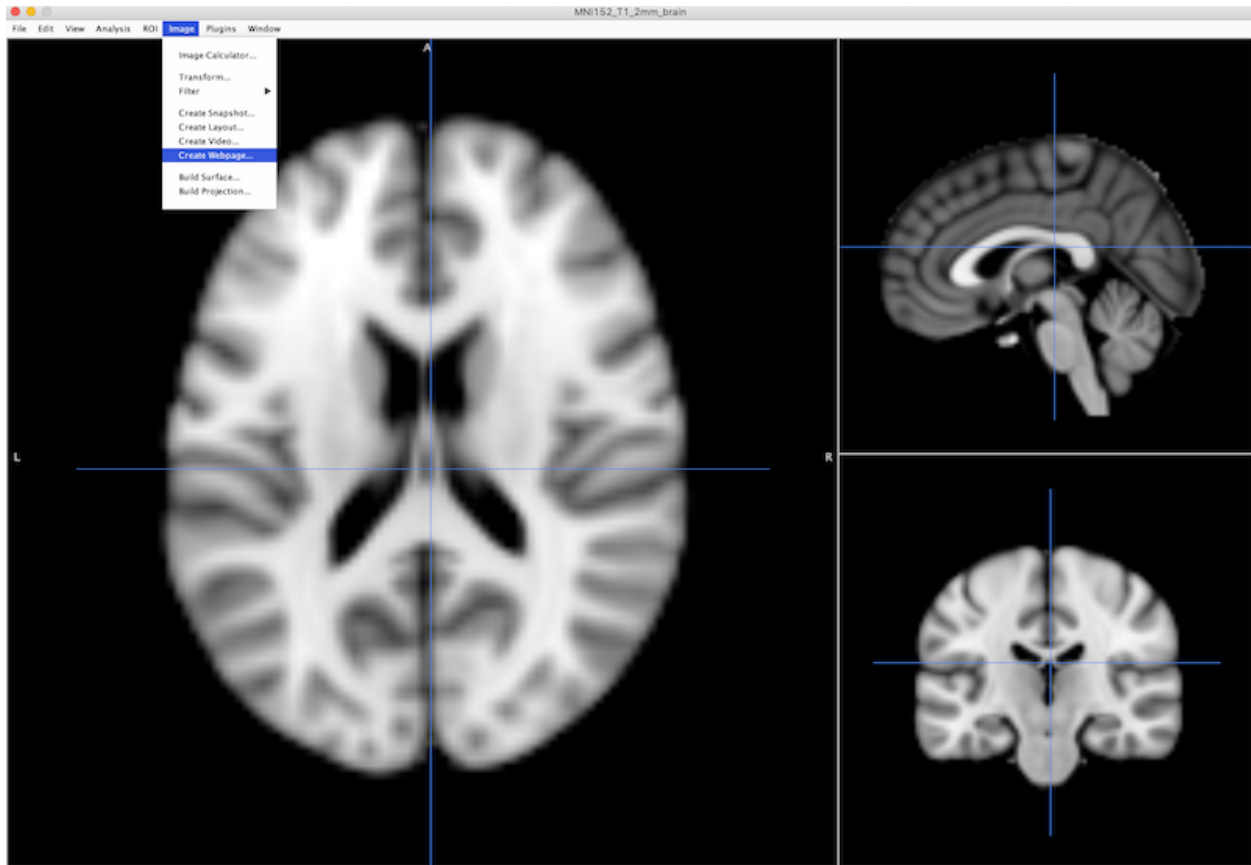
> **Warning:** As of July 25, 2019: Depending on the way the input data is stored, Left and right can be switched in the HTML viewer. See *Ensure Standard Right-to-Left Orientation in NIFTI Header* for more information.

- Choose *Create* to get a default viewer like this. Note this is displayed as a webpage, and has several useful controls along the bottom.

  - You can drag the cursor and see your current x,y,z position and intensity. You can also use the + and - buttons next to x,y,z along the bottom to move one voxel at a time.

  - On the bottom right is a color table control (currently set to greyscale).

  - *Swap View* will change which orthogonal plane appears in the larger panel on the left.
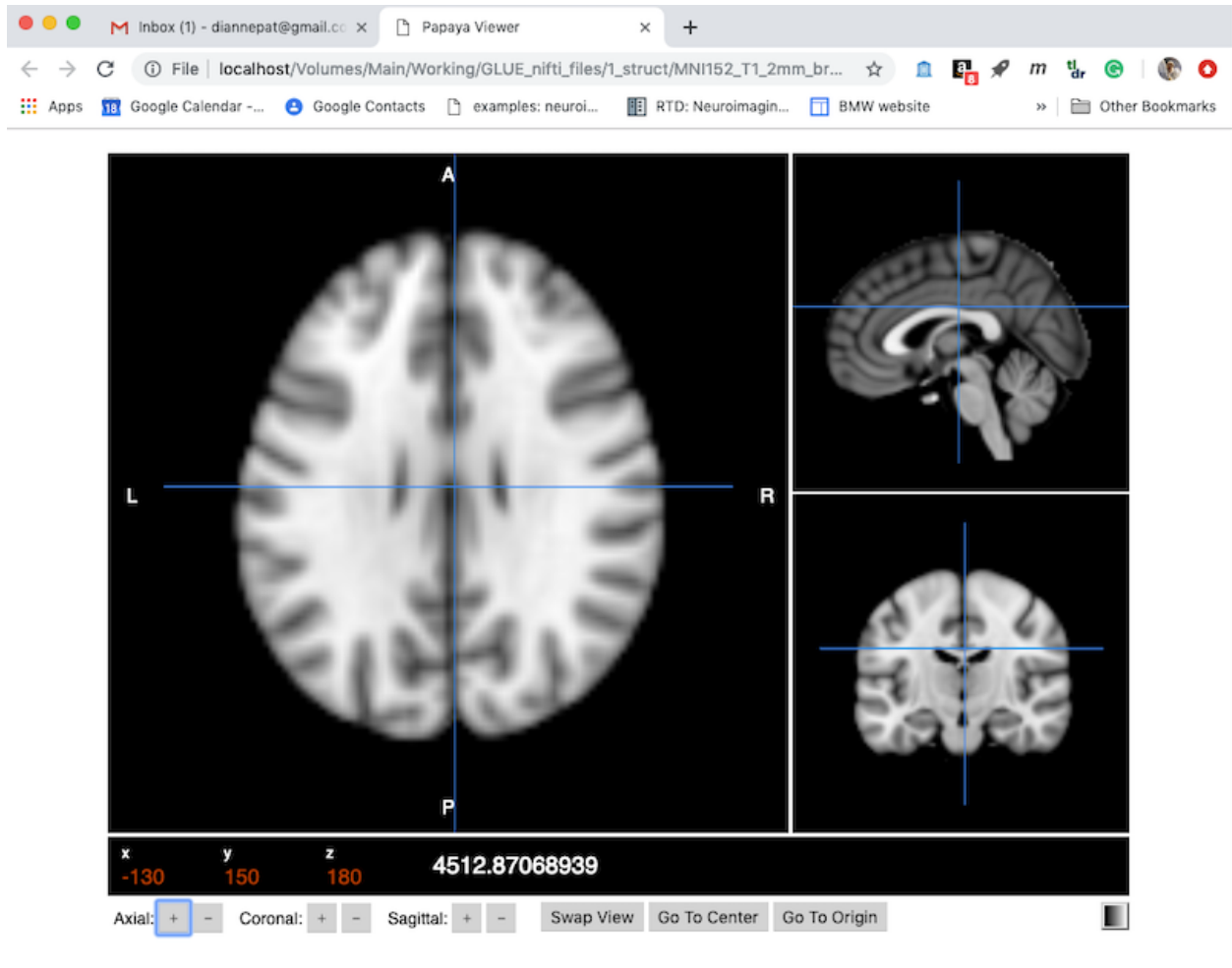
- Click *More Options* (upper right) to expand the choices to what you see below:

- You can make a title, e.g., "Here is a lesion for sub-199" and a footnote (e.g., "created with Mango").
- *Kiosk mode* hides the menu. If you choose *Main view only* then it only shows one panel, though you can still swap views.

- You can build a surface in Mango: *Image Build Surface* and include that in the webpage:
    - Choose "include surfaces" (upper right) to display a 3D object then "swap view" (lower right on webpage) to make the render the main panel and then move around. https://github.com/rii-mango/Papaya/wiki/User-Interface-Features
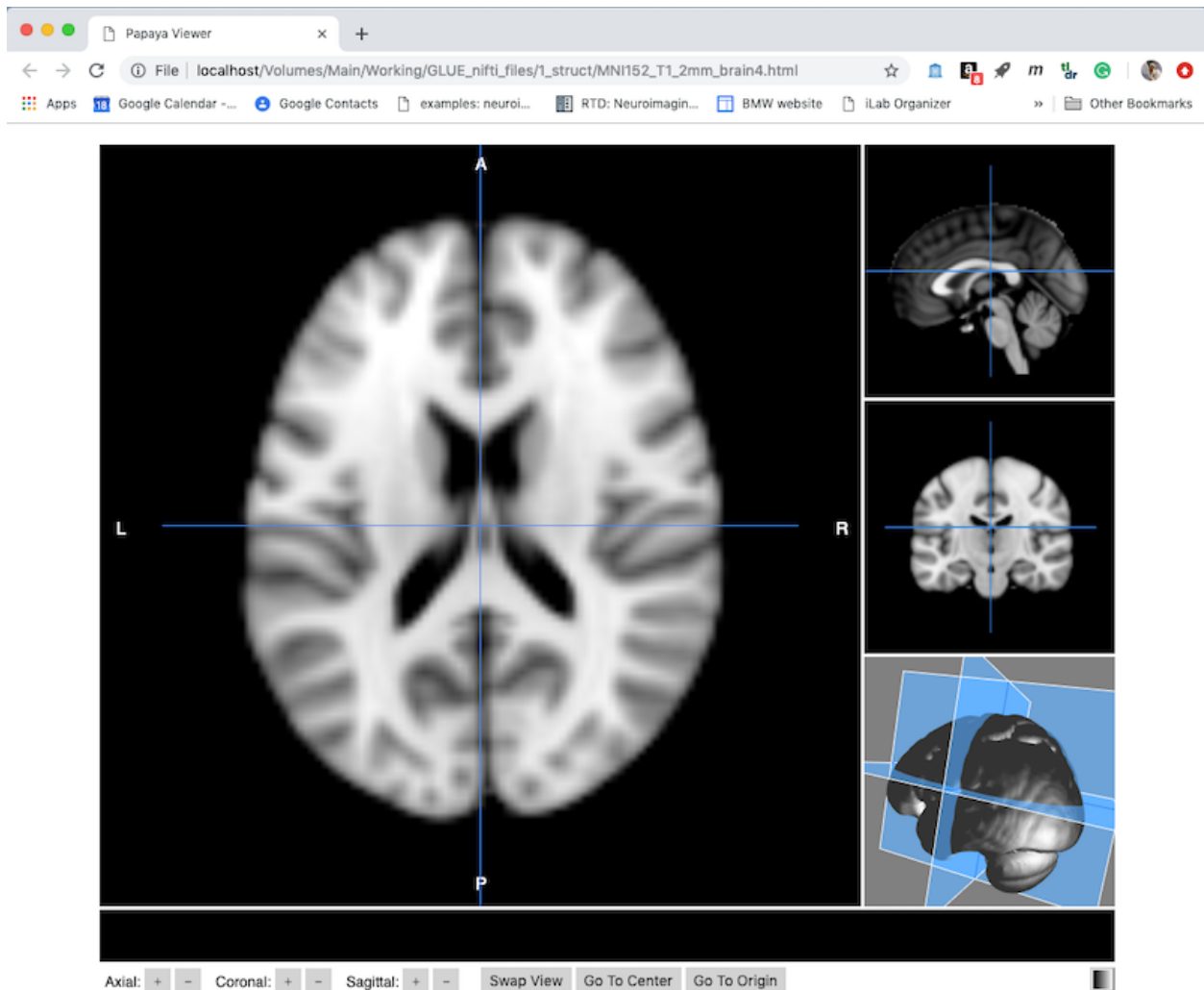
CHAPTER 13

SPM

Maintainer: Dianne Patterson dkp @ email.arizona.edu

Date Created: 2018_06_14

Date Updated: 2018_11_17

Tags: Neuroimaging software, SPM12, Matlab

OS: Mac, Linux or Windows with Matlab (no toolboxes necessary)

Among his other tutorials, Rajeed Raizada provides some excellent interactive Matlab tutorials for understanding how SPM analysis works: Standard fMRI Analysis See especially hrf_tutorial.m math_of_convolution.m and design_matrix_tutorial.m

13.1 Batch and Scripting

- Use the SPM GUI to create a batch script that you wish to apply to other subjects.

- See *SPM Lesion Normalization* for an example of creating a batch job

- Insure that the batch job runs properly.

- If the file names are the same for all subjects, this will make your life easier. If not you can link some simplified names to your subject-specific names. Again, see *SPM Lesion Normalization*.

- In the batch editor, choose *File Save Batch and Script*, and type in a name (e.g., nrm1). This will save two m files, e.g., *nrm1.m* and *nrm1_job.m*. Put these in a directory in your Matlab path.

- If you want to load the batch up in the batch editor again, start the editor and load the *job.m* file

- Look at the job file first. You see repeated calls to matlabbatch. Each call is followed by a number, e.g., {1} for the first batch module you set up, {2} for the next one, etc.

- There are several spots where the path to your data is specified. We'll want to change these. For example, the first line of my nrm_job.m is:

```
matlabbatch{1}.spm.spatial.preproc.channel.vols={'/Volumes/Main/Working/
↪LesionMapping/DataC8/sub-001/T1w.nii,1'};`
```

- We need something more generic than */Volumes/Main/Working/LesionMapping/DataC8/sub-001/T1w.nii,1*. So I add this line at the top of the script to define a variable for the anatomical image name:

```
T1='T1w.nii'`
```

- Now I modify the line in question to look like this (substituing my generic variable for the full path to the file for sub-001:

```
matlabbatch{1}.spm.spatial.preproc.channel.vols = {T1};
```

- Define any other image files using the same idea, and substitute your new image names with these variables throughout the script. If you list multiple file variables, then separate them by commas, and insure they appear in a column, like this:

```
{
  T1,
  lesion,
  T1_brain
}
```

- Now we are going to change the other file (e.g., nrm.m) completely. Our goal is to define the data directory path and the subject folder names (This assumes all of your subject data is in subject directories in a single folder). Then we will loop through the subjects, cding to each subject directory, running our batch job, and then going back up to the dataDir before starting on the next subject:

```
% This is a comment: clear any existing variables
clc; clear all
% Insure SPM is ready to go
spm('defaults', 'FMRI');
% Define the data directory path containing your subject folders
dataDir = '/Volumes/Main/Working/LesionMapping/DataC8/Test/';
% Define a list of subject directories (probably longer than this)
subjList={'sub-030','sub-031','sub-033'}
% for each subject in subjList, put the subject in a variable called "subj"
for subj = subjList
    % display the subject (not necessary, but helpful)
    disp(strcat('Subject:   ',subj));
    % clear matlabbatch variables (we don't want any leftovers contaminating this
↪run)
    clear matlabbatch;
    % cd into the subject directory.
    cd(char(subj));
    % call the job nrm1_job.m
    spm_jobman('run','nrm1_job.m')
    % When finished, go back up to the dataDir
    cd(dataDir)
end;
```

- That's it. Test your script by typing the name of the file we just modified (using a small subject list):

```
>>nrm1
```

Learn Unix Online with Jupyter lab

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2018_09_12
Date Updated: 2019_08_21
Tags: UNIX

You can use the Unix command line to interact with your mac, with any supercomputer or any linux system. Learn three important commands and build on those.

Bash (born again shell) is a very common shell (command line interface). You can even download Bash for Windows! Here I will walk you through some basic commands using an openly available online resource called Jupyterlab. JupyterLab is evolving quickly, so they periodically change the icons and move stuff around. I'll try to keep up with these interface changes (You can check the **Date Updated** above to see how stale the information is). If the interface has changed while I wasn't paying attention, forge ahead. . . everything should be the same once you figure out how to open a terminal window.

Go to Jupyter.org/try, which includes a free online unix terminal.
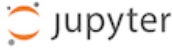
You will see an interface like this:

Your jupyterlab terminal displays a Bash command prompt where you can type commands. Your command line will probably look a lot like mine: jovyan@jupyter-jupyterlab-2djupyterlab-2ddemo-2d9fn107tw:~$

In the rest of this tutorial, commands that you should type are set off like this:

```
ls
```

Throughout the tutorial, you will also see questions in **bold**. You should try to answer these questions. You are now ready to try the basic commands explored below.

Fig. 1: Click **Try JupyterLab**.
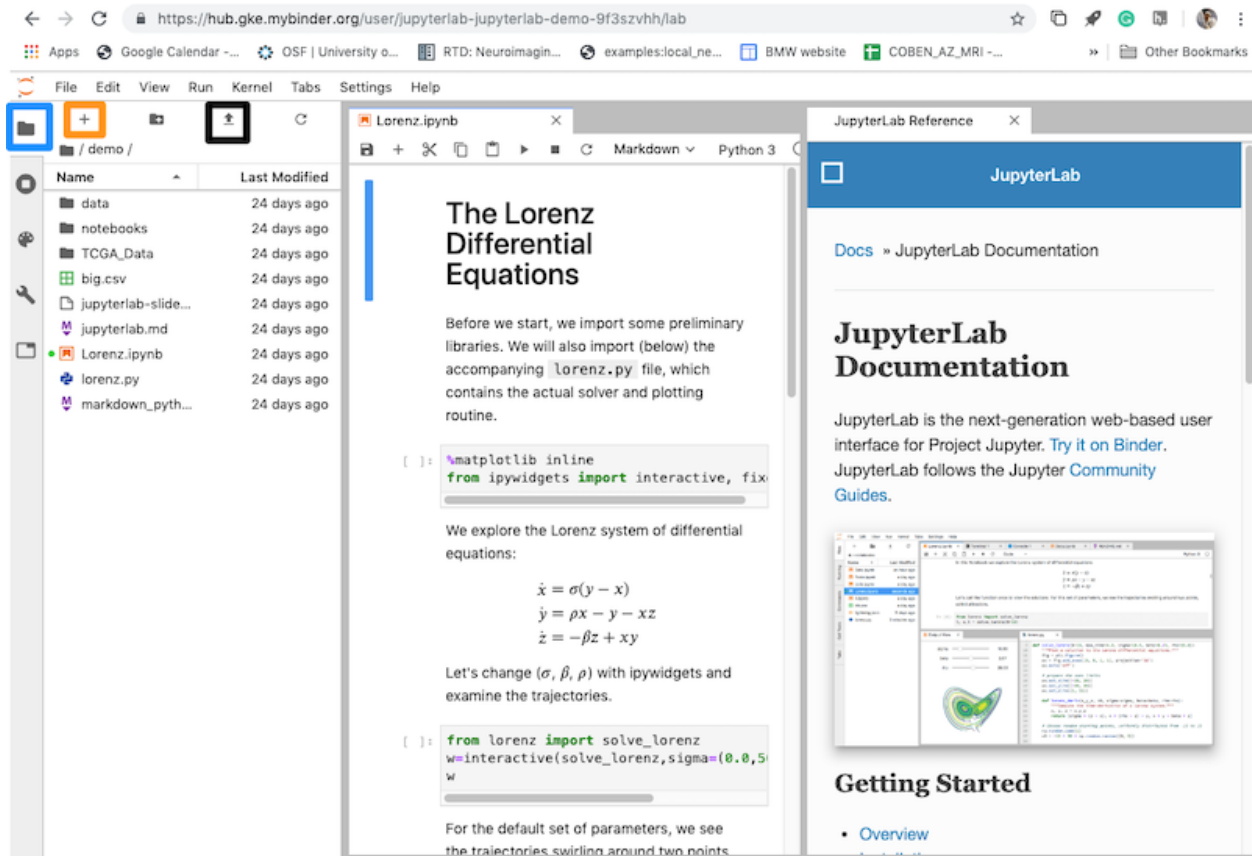
Fig. 2: Note the icons on the upper left. **Blue Box**: This is the **Files** icon. Click it to see what happens: You should see it toggle the lefthand pane that lists the available files on the site. You will need it later. **Orange Box**: This is how you add a new launcher. You will need this in just a moment. **Black Box**: This is how you upload files. Keep it in mind for later. **Add a new Launcher** by clicking the + sign in the orange box.
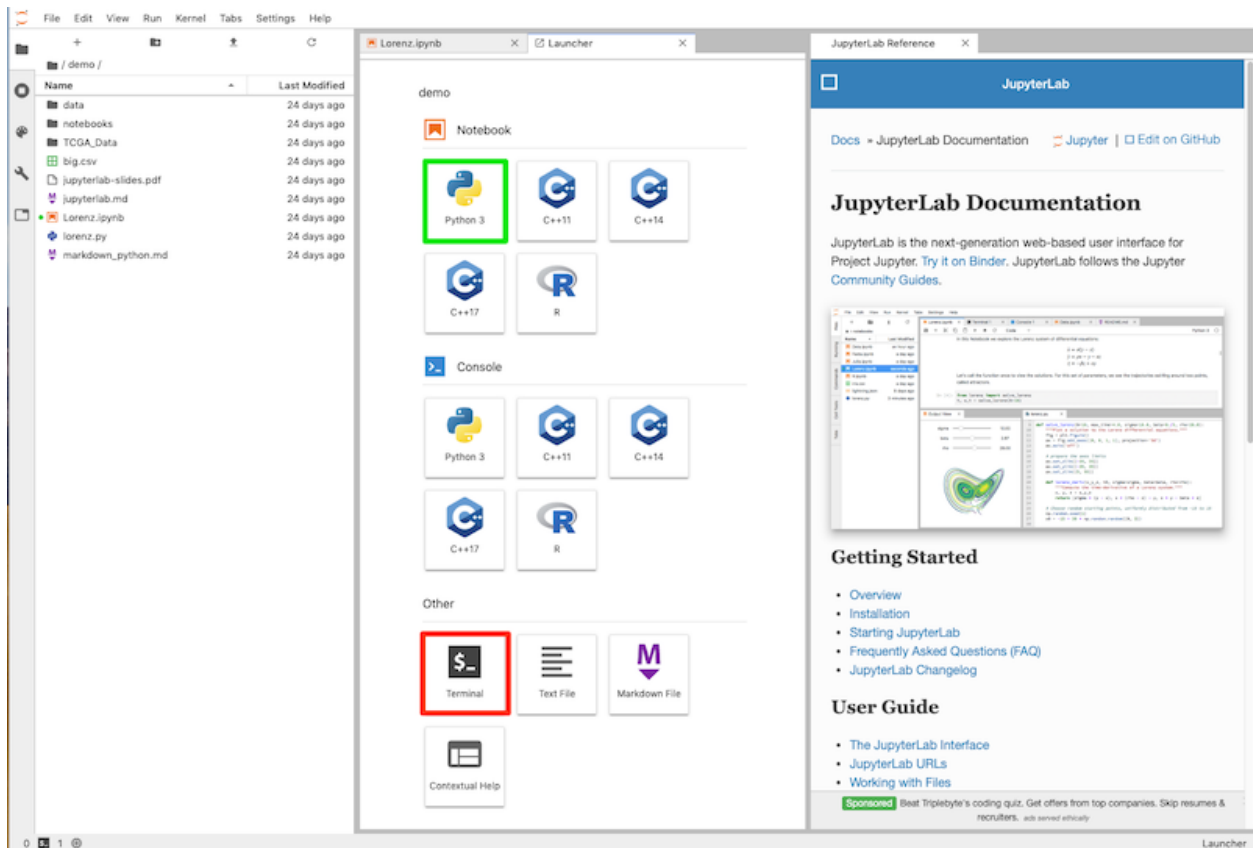
Fig. 3: On the Launcher tab, click **Terminal** (red box) from the **Other** category.

14.1 Three Commands to Rule them All

Table 1: Command Meanings

Command	Meaning	Example
pwd	Print Working Directory	pwd
ls	List	ls
cd	Change Directory	cd /

Find out what directory you are working in:

```
pwd
```

List the contents of this directory:

```
ls
```

Go to the root directory (there is a space between the command `cd` and the destination `/`):

```
cd /
```

List what is in the root directory:

```
ls
```

14.1.1 Flags and Arguments: Examples with ls

Goal: You have just used three different commands. Learn to use command flags and arguments to expand your options.

At the prompt, one types commands, optionally followed by flags and arguments. Always start a command line with a command. A flag is a dash (-) followed by one or more letters that add options to that command.

list all: list all files and directories, including the hidden ones that start with a dot:

```
ls -a
```

list long: distinguishes directories, files, and links; and provides ownership and permission information:

```
ls -l
```

list all, long:

```
ls -al
```

An argument is what the command operates on, usually a file or directory.

Give `ls` the argument `/etc` to see a list the files and subdirectories in that directory:

```
ls /etc
```

14.2 Get Help

Goal: Learn how to find help for any command.

Unfortunately, the Jupyterlab terminal does not offer man (man=manual) pages, but you can google `man` and any command to find help on that command (on most real systems, you can type `man ls`, for example, to learn about the ls command).

Google `man ls` to see all the flags for `ls`.

14.3 Directories, Links and Files

Goals: Learn to distinguish different types of items listed by `ls`

From the root directory, type:

```
ls -l /etc
```

To distinguish directories from links (aliases) and files, look at the 10 characters at the start of each row, e.g., `drwxr-xr-x`. The first character `d` in this example tells us the item on this row is a directory. Next, the character string tells us the user, group and other permissions (as in the table below). For right now, we care about the first character, the `d`, that indicates this is a directory. You will also see `-` to indicate that an item is the default, a file; and `l` to indicate that an item is a link.

Table 2: Command Meanings

File Type	User permissions	Group permissions	Other permissions
	read write execute	read write execute	read write execute
d	r w x	r - x	r - x

Type `cd` with no arguments, and bash takes you home:

```
cd
```

Another useful command is `file`, type the `file` command followed by the argument `*` which means *everything/anything*:

```
file *
```

Unix tells you something about each file and directory in the current location.

14.3.1 Directories

Goal: Find out where you are and move around in the directory structure.

Use `pwd` to determine your current location. The location is displayed as a path.:

```
pwd
```

Return to your home directory using this trick (`cd` with no argument):

```
cd
```

Make a new directory named **test**:

```
mkdir test
```

How would you change directory to go into the test directory? Do so.

To find out if anything is in the test directory, type:

```
ls -la
```

Every directory contains at least two hidden directories . and ..

. means *here*

.. means *one level up*.

Use cd to go up one level, like this:

```
cd ..
```

What directory are you in now?

Does the following command do anything? Why?

```
cd .
```

These commands you've been typing are all files on computer. Where are these command files? To find a command on your system, use which:

```
which pwd
```

pwd is in /bin with many of the other commands on the system.

Change directory to /bin

List the contents /bin

What kind of file is pwd?

```
file pwd
```

You can use a command called history to list the commands you have run! Usually, history is set to remember between 100 and 1000 of your most recent commands:

```
history
```

Pick one of the commands and run it again from your history using the exclamation point (a.k.a. bang) and the number of the command in the history, e.g.,:

```
!4
```

14.4 Summary of Main Points

- You learned some commands:
    - pwd (print working directory);
    - ls (list)
    - cd (change directory)
    - file (display more information about the type of file)

       – mkdir (make a directory)

- You learned about flags and arguments.

- You learned that files and directories can be hidden by naming them with an initial `.`

- You learned that there is a manual page for most unix commands. Normally, you can access the manual page at the command line. Sadly, the Jupyterlab terminal does not have a man page, but you can google the man page you want. If you are in a shell that displays man pages, then you can use the down arrow to see more of the page, and you can type `q` to quit the man page.

- You learned that you can view the history of commands you have typed, and rerun them.

14.5 What's in that File?

Goal: Learn to view and edit the contents of a file.

Go to your home directory:

```
cd
```

Use `cat` (cat=concatenate, but is often used to view files) to view the *README.md* file in your home directory. Let's be lazy and use *tab completion* so we don't have to type the whole name:

```
cat REA
```

Now hit the `tab` key to complete the command. If the completion looks correct (it should), then hit enter to execute the command.

Let's create a bash script. Start a new launcher if you need to (+ in orange box, upper left).

Click the Text Editor button.
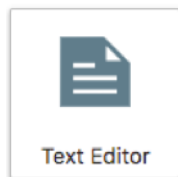
In the editor, enter these 4 lines:

```
#!/bin/bash
echo "what is your name?"
read answer
echo "hello $answer!!, pleased to meet you."
```

- Rename the file to *hello.sh* (you can right-click the name of the file in the tab or in the file browser on the left).

- Because you renamed the file with the .sh extension, the editor now understands that this is a bash script, and it uses syntax highlighting to help you distinguish commands, variables and strings. There are 2 ways to run your script.

Try this one:

```
bash ./hello.sh
```

Later, you will change the permissions on the file, and that will make running it even easier.

14.5.1 Summary of Main Points

- You have learned to use tab completion to reduce typing.

- You've used `cat` to view the contents of a file.

- You've used a code editor to type in text and see highlighting appropriate for the syntax of the programming language you are working in (you were writing in bash here).

14.6 Permissions

Goal: Learn to read file permissions and understand how and why to change them.

Remember those permissions we saw earlier in the table? Permissions are meant to protect your files and directories from accidental or malicious changes. However, permissions are often a significant obstacle to sharing files or directories with someone else and can even affect your ability to run scripts.

Let's see the permissions on *hello.sh*:

```
ls -l
```

- They look like this: `-rw-r--r--`.

- This says hello.sh is a file `-`.

- The user can read and write *hello.sh*, but cannot execute it `rw-`.

- Other people can only read *hello.sh* `r--`, but cannot write to it or execute it.

- No one has `x` (execute) permissions on *hello.sh*.

Try to execute the script:

```
./hello.sh
```

It does not work.

Change permissions on *hello.sh* and make it executable by user, group and other:

```
chmod ugo+x hello.sh
```

This changes the mode of *hello.sh* by adding `x` (executable) for everyone (`chmod a+x hello.sh` will also work. a=all):

```
ls -l
```

What are the permissions on hello.sh now?

We are going to run hello.sh again:

```
./hello.sh
```

hello.sh should ask you your name and then greet you.

Let's use cat (concatenate) to read *hello.sh*:

```
cat hello.sh
```

Now use chmod to remove read permissions for user, group and other on *hello.sh*:

```
chmod ugo-r hello.sh
```

- Look at permissions on *hello.sh* now
- Use `cat` to read the contents of *hello.sh* again. You no longer have permission to read it.

How do you use chmod to make hello.sh readable again?

14.6.1 Summary of Main Points

- You have learned that permissions are important.
- You learned the command `chmod` for changing the permissions of user, group and other.
- A constant annoyance on unix machines you share with other people is that files and directories get created by one person, but the permissions don't allow someone else to work with the files. So it is really important to look at the permissions when you get a `permission denied` message.

14.7 Standard Unix Directory Structure

Goal: Learn about the important directories you can expect to find on any unix system. Learn to better understand navigating those directories by understanding the path, especially whether it is relative or absolute.

Unix directories are organized into a tree with a root (/) at the top and branches underneath containing directories and files
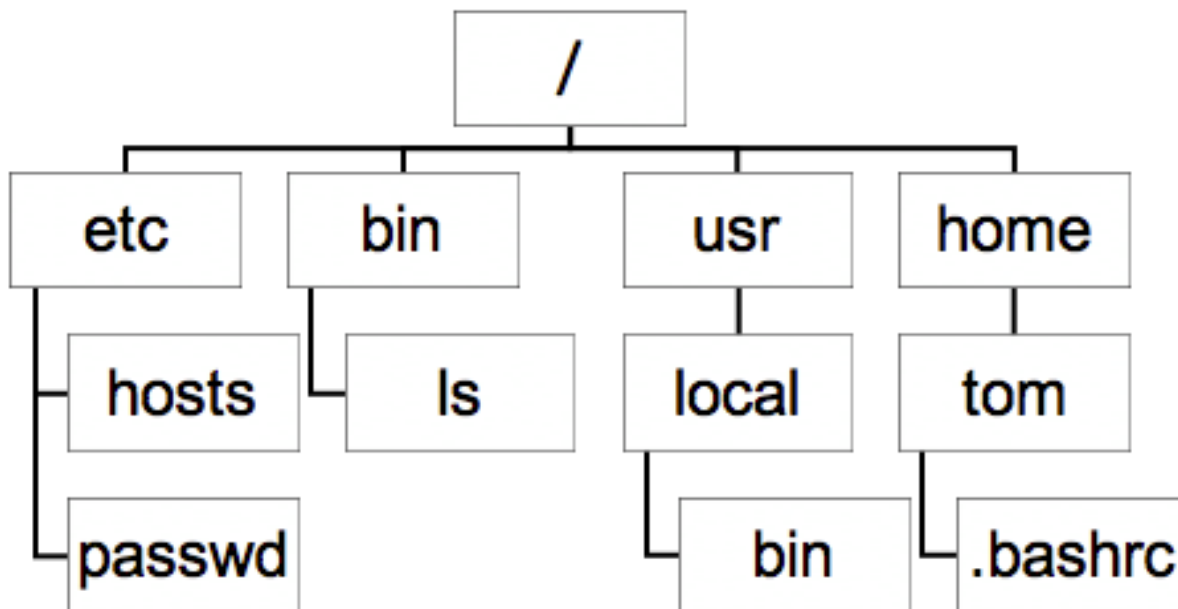
14.7.1 Important Directories

- `/bin` where many built-in unix commands are kept
- `/etc` where system profiles are kept; e.g., hosts, passwd, …
    - *hosts* is like a telephone book for computer IP addresses and names.
    - *passwd* stores the encrypted passwords of all users.
    - Only the root user has the ability to modify files in `/etc`.
- `/usr/local/bin` where non-built-in programs often get added
- `~` is your home directory: where your profiles are kept and where your command line starts

How would you look at the contents of the hosts file?

14.7.2 The Path

- A path is a recipe for navigating the directory tree. You must follow the branches.
- When you use `cd` you must give it a path as an argument.
- The shell has a list of paths where it looks for commands. This list is refered to by the variable name `PATH`

Use echo to display the contents of the variable PATH (Variables can be identified with an initial $ sign):

```
echo $PATH
```

What is the first directory in the shell's path? (directories in the path are separated by colons)

If commands with equivalent names (e.g., ls) exist in different places in the shell's path, the first one will always get executed. If you install new software on your unix system, and then a command stops working as expected, you should check to see if a command of the same name has been added by the new software. That is, ensure you are running the command you think you are running, rather than a new one with the same name. For example, if a command called `ls` was overriding your `/usr/bin/ls`, you could find out by typing:

```
which ls
```

14.7.3 Absolute and Relative Paths

- An **absolute path** specifies the entire path starting with root. For example, the absolute path to the `.bashrc` above is `/home/tom/.bashrc`. If I am in the directory home already, then I may prefer to use the **relative path**: `tom/.bashrc` just because it is less typing.
- **What is the absolute path to the file passwd?**
- **If I am in the directory /etc, what is the relative path to the file passwd?**

14.8 Actions: cp, mv, rm and wildcards

Change directory to your test directory. Under test we'll create a structure like this:
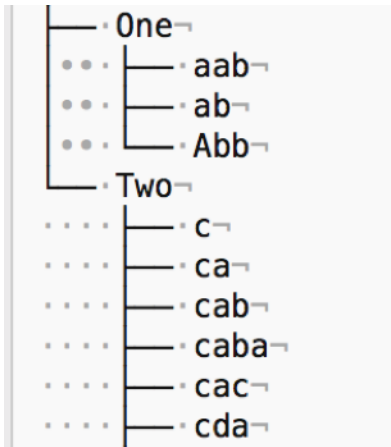
Create the directories *One* and *Two*:

```
mkdir One Two
```

Create files in each dir using `touch`:

```
touch One/{aab,ab,Abb}
touch Two/{c,ca,cab,caba,cac,cda}
```

Ensure that everything has been created correctly by using `ls` recursively to look at subdirectories:

```
ls -R
```

Use `echo` and the redirect symbol > to add content to some of the files. This will help us keep track of how the files are getting moved and changed:

```
cd Two
echo "what is a caba?" > caba
echo "I am the cutest" > cute
```

You never created a file named *cute*. Do you have one now?

Copy or move a file to another file:

```
cp ca caba
```

What's in caba now?

```
mv cute caba
```

- **What is in caba now?**
- **Where is cute?**
- **How are copy: cp and move: mv different?**
- **Why do cp and mv require two arguments?**

Move a file up one level in the directory structure:

```
mv caba ..
```

- **Where is caba now?**
- **How would you copy caba into the directory One?**

- **How many caba files now exist?**

- **What is the absolute path to each caba file?**

Copy or move a directory:

```
cp Two One
```

Did it work? (Use `ls -R` to see what happened). If you are inside the directory *Two* when you type this command, bash cannot find the directory *Two*. If you are above the directory *Two* (i.e., you are in test), it still fails. To copy a directory, you must issue the command recursively. Try this:

```
cp -R Two One
```

How is copying a directory different than copying a file?:

```
mv Two One
```

How is moving a directory different than copying it?

14.8.1 Rename a Directory

`mv` works differently when the target directory (i.e., the 2nd argument) does not exist:

```
mv Two Three
```

- **What happened to Two?**

- Move the directory *Two* from inside of *One* to test. **What commands did you use?**

14.8.2 Learn about Wildcards

Go to the directory *Three*.

Use wildcards: `?` = one character; and `*` = any number of characters):

```
ls ca?
```

Which files are listed? Why?:

```
ls c*
```

Which files are listed? Why?:

```
ls c?
```

Which files are listed? Why?

14.8.3 Learn about remove `rm`

Remove *ab* from the *One* directory (this command assumes you are in the One directory):

```
rm ab
```

Remove a directory (**How is this different than removing a file? Where do you have to be in the tree for the command to work properly?**):

```
rm -r Three
```

There is NO UNDELETE on unix. Be careful with `cp`, `mv` and `rm`.

14.8.4 Permissions Again

Now that you have created a tree of files and directories, you may want to control who can get to it. To make sure a directory and all of its files and subdirectories are readable by everyone, do this:

```
chmod -R ugo+rwx One
```

This recursively `-R` changes permissions for *user*, *group* and *other* (ugo) to add read write and execute permissions (rwx) to the directory *One* and all of its contents. The recursive flag `-R` (and sometimes `-r`) occurs for many commands.

14.8.5 Summary of Main Points

- You have learned to move, copy and remove files and directories.
- Files and directories cannot be retrieved after being removed, so BE CAREFUL: Use `cp` instead of `mv`, to reduce the chance of losing things.
- Consider making a test directory like this one to make sure you know how the commands are going to affect your files.

14.9 Glossary of Commands

- bash
- cat
- cd
- chmod
- cp
- echo
- file
- history
- ls
- man
- mkdir
- mv
- pwd
- rm
- which

To learn about creating your own script directory and modifying the path, see *More Unix: Install Neuroimaging Tools*.

14.10 Other Unix Resources

- Check out excellent tutorials and videos from Andrew Jahn on unix commands for neuroimagers.
- Research Bazaar Unix Introduction
- *Bookmarked Computational Resources*

More Unix: Install Neuroimaging Tools

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2019_08_21
Date Updated: 2019_08_21
Tags: UNIX

We want to pull together things you have learned to make you more independent. It is one thing to run commands on a system someone else has set up for you. But it requires a bit more understanding to set up things yourself.

15.1 Install FSL

I am going to illustrate with FSL scripts, because I know that best and there are a number of scripts available on this site that rely on FSL. This means you will want to install FSL. Installing FSL can be a big job. Be prepared to follow the instructions on the FSL site carefully. It is possible to have problems even if you follow the instructions (e.g., `imcp`, `immv` and several related commands sometimes fail to install correctly). Join the forum, try the install, take some breaks. If you can't make it work, send a detailed description of the problem to the FSL list (your operating system, what you tried, what error messages you get). Be persistent, keep notes, and forgive yourself your errors. Once FSL is installed, you can continue with the steps described below.

15.2 Set up a Tools Directory

In this section, you will download some scripts and make sure they are executable and in your path.

Step 1: Go to your home directory and create a subdirectory called tools:

```
cd ~
mkdir tools
```

Step 2: Go to the Prepare T1w Images section of these pages. Click the links for each of the three scripts (`prep_T1w.sh`, `fsl_anat_alt.sh` and `optiBET.sh`), and download each one of them into your tools directory.

Step 3: Make sure the permissions on those scripts will allow anyone to execute them:

```
cd tools
chmod a+x *.sh
```

Step 4: Find out which shell you are running (e.g., bash, tcsh, csh, zsh etc). You are probably running bash (since that is the default shell on most machines). You can check like this:

```
echo $SHELL
```

Step 5: Find or create your shell configuration file. There should be a shell configuration file in your home directory (`.cshrc` for csh, `.tcshrc` for tsh, `.bashrc` for bash). The configuration file will be hidden (i.e., it'll start with a `.`). You may find you have configuration files for several different shells, or none at all.

First make sure you are in your home directory (not in tools). Then display all hidden files to see what configuration files you have:

```
cd ~
ls -la
```

If you don't have any configuration files, then all of your settings come from a shared configuration file in the `/etc` system area. Don't worry. You can simply create your own, e.g.,:

```
touch .bashrc
```

You can find minimal configuration file examples on the internet, e.g., minimal .bashrc and .bash_profile

> **Warning:** Configuration files from different shells are not interchangeable. The way the paths and aliases are created can differ considerably. That said, `bash` and `zsh` are very similar to each other and should be able to use the same configuration statements. `tcsh` and `csh` are also similar to one another and pretty interchangeable. However, tcsh and csh are very different from bash and zsh.

Step 6: View your configuration file. You don't need an editor to view your configuration file. You can use `cat`, e.g.,:

```
cat .bashrc
```

Step 7: If there is something in your configuration file already, you don't want to mess it up. Make a copy right now, e.g.,:

```
cp .bashrc bashrc_orig
```

Step 8: Edit your configuration file. You will want to change and test your configuration file and you need an editor to do that. There is always an editor called `vi` available, but it is really old-fashioned and difficult to use. You should learn to work with it someday, but not now. On the mac, you should have `textedit`, and on Ubuntu you should have `gedit` by default. You may want to install an editor like atom. Once you have found or installed your editor, you can open the configuration file of choice with it, e.g.:

```
gedit .bashrc
```

Step 9: Add a line to your configuration file to ensure the operating system will look for executable files in your tools directory. This is an example of a bash path (You can look for examples online for other shells).:

```
PATH=$PATH:~/tools
```

This says: the path shall contain everything previously defined for the path AND the directory `tools` in my home area.

Save your changes.

Step 10: Did you break it? From your home directory, source your configuration file, e.g.,:

```
source .bashrc
```

If nothing happens, you did it right! If there are errors, read the messages! As soon as the system finds an error in your configuration file, it stops trying to understand any subsequent statements. If you have a backup, you can source that to see if it had the same problem (or if it was indeed your fault), e.g.,:

```
source bashrc_orig
```

Step 11: Can the operating system find your scripts?:

```
which fsl_anat_alt.sh
```

We hope the operating system will find the script you put in `~/tools`. If so, you did it! If the operating system says `fsl_anat_alt.sh not found`, then check these things:

- Are the scripts executable?
- Did you save the changes to your configuration file?
- Did you source the configuration file?

Warning: If you have several shell windows open (e.g., several bash windows, for example), sourcing your `.bashrc` in one of your windows only affects that window! If you open a new bash window, it will automatically source the `.bashrc` when it starts up. But, windows that were already open do not know about changes in the `.bashrc` until you force them to by typing `source .bashrc`.

If you got through that, congratulations! You have learned a really important skill for taking control of your computing environment.

HPC for Neuroimaging

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2019_07_24
Date Updated: 2020_04_06
Tags: UNIX

16.1 Useful Links

- HPC Introduction

- HPC Documentation University of Arizona High Performance Computing Documentation

- HPC Account Creation

- OOD Open On Demand Dashboard to interact with HPC resources.

- Check Disk Quota and Usage

16.2 Introduction

This page provides a tutorial for the neuroimaging community about getting started with the HPC (High Performance Computing) cluster at the University of Arizona. The focus is on issues that are likely to be most relevant to neuroimaging. Interfaces for interacting with the HPC are much easier to use than they were years ago, and they are evolving rapidly. Look at **Date Updated** above to ensure that you are looking at reasonably current documentation.

The HPC (High Performance Computing) is a very large cluster of linux machines at the University of Arizona. You can interact with the HPC at the command line or through a graphical user interface, but to use it effectively you should have some basic knowledge of *unix*.

Many big universities have similar HPCs to support research. In additon, the NSF funds other computing services for scientific research, for example *Cyverse* and Xsede, both of which are nationwide. For now we'll describe the U of A HPC.

16.3 PROS

- Free to U of A researchers, so you can run a lot of image processing using your monthly time allocatation on the HPC.

- If an image processing tool requires huge amounts of RAM (e.g., maybe some GIFT processing?), the HPC can provide that.

- The HPC provides lots of high-powered NVIDIA GPUS which can really speed up our neuroimaging tools IF they are GPU-enabled.

- The HPC can run Singularity containers.

- The HPC can scale up to run lots of processes in parallel. For example, even though Freesurfer's recon-all will still take 12-24 hours to run, you could run it on dozens of subjects at the same time.

- Once you are familiar with the HPC, if it is not sufficient for your needs, you could request access to XSEDE. Again, you'll get a lot of processing time for free.

16.4 CONS

- Data must be deidentified before you upload it to the HPC (the HPC is not HIPAA compliant). See *Deidentifying Data*.

- You have to move your data to and from the HPC for processing. See *Transferring Files*.

- The HPC is meant for processing, not storing your data:

  - You get ~200 GB of space under /extra (This is not backed up).

  - You can get up to 1 TB of space on xdisk for up to 150 days (this is quick and free).

  - You can purchase storage space, for about $40 per Terabyte per year.

  - The storage options will be changing during 2020

- The CPUs are not that fast, so do not expect processing of an individual subject to be faster than your local machine (I find it is slower than my mac pro).

- You do not have administrative privileges:

  - You are not allowed to run Docker

  - You cannot build a Singularity container from a Singularity recipe file on the HPC.

    * See Sylabs Cloud Builder for building Singularity containers from a recipe file. I have built a 1.6 GB image in about 7 minutes. It is free, and if it'll handle the size and build time for your container, then it is really fast and easy.

    * If your recipe is too big for the Sylabs cloud, see *Singularity on Cyverse* for a solution.

- GPU processing is complex to set up and NOT a universally available panacea for slow processing.

  - The software tool has to be designed to use GPUs and that is only practical for certain types of operations.

  - For example, GPUs don't help at all with Freesurfer's recon-all but can help with FSL's probtrackx2, Bedpostx and Eddy, all three of which provide gpu-enabled versions.

16.5 Tutorial Introduction

Below I lay out a set of steps for you to try, in what I hope is a reasonable order. Please let me (dkp @ email.arizona.edu) know if any of it is confusing or wrong. Don't assume it is just you! If you get confused by the sequence of steps or the wording seems ambiguous or misleading, then other people will have the same experience and the tutorial can be improved, so I need to know. Thanks.

16.6 Sign up for an Account

Note: If you are a PI, you must follow the steps in this section to **create an account** and **add users**. You do **NOT** have to sign on to the HPC or learn Unix or anything else, but you must create an account to get an allocation of HPC resources for your lab, and you must add any staff or students who you want to have access to that allocation. See PI Account Instructions.

- Everyone must go to the HPC Accounts Creation page for instructions on signing up.

- It'll take under an hour for your home directory to be created, but you won't have access to disk space on /extra for a while longer. The HPC will send you an email when your home directory is ready.

16.7 Sign on to OOD

Once you have an account: log on to OOD (Open On Demand). Be persistent if this is your first time (it may take a few trys to be acknowledged as a bonafide user). OOD provides a dashboard interface to the HPC system:
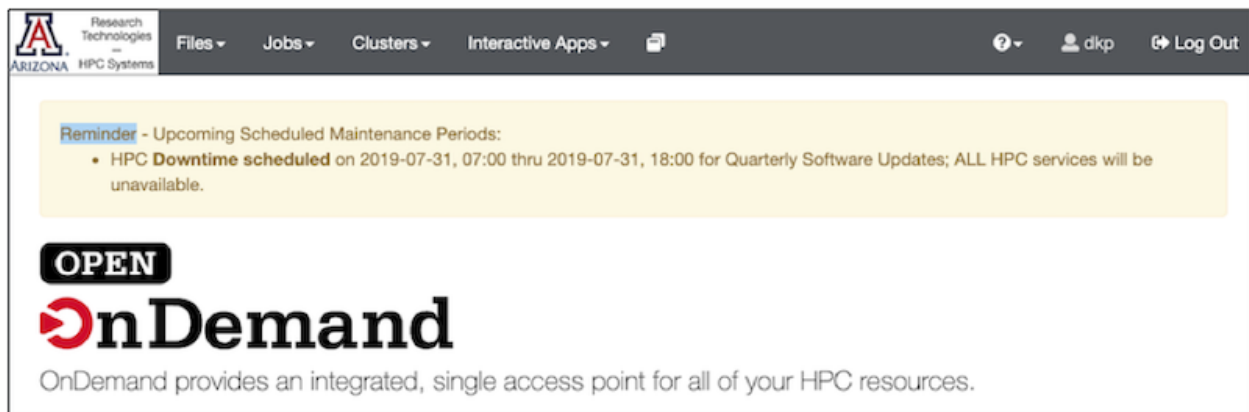
Fig. 1: The **OOD Dashboard**: Along the top is a grey bar with the options: Files, Jobs, Clusters, Interactive apps.

Click each one to see what it offers. Briefly:

- **Files** offers you a graphical user interface for exploring your directories.

- **Jobs** allows you to see the status of running batch jobs and to delete them. We'll get to this later in the section on *batch jobs*.

- **Clusters** allows you to open a command line (a.k.a shell or terminal window) on one of the machines (e.g., ocelote or el gato). To use this, you need to know some *unix commands*.

- **Interactive Apps** allows you to open a graphical desktop, Jupyter notebook or R-studio session.

We will focus on **Files** for now. On the top left, click **Files**. You will see options to view your home directory (at least). **Select your home directory**. You should see something like my home directory displayed below (but you'll have less stuff in yours).
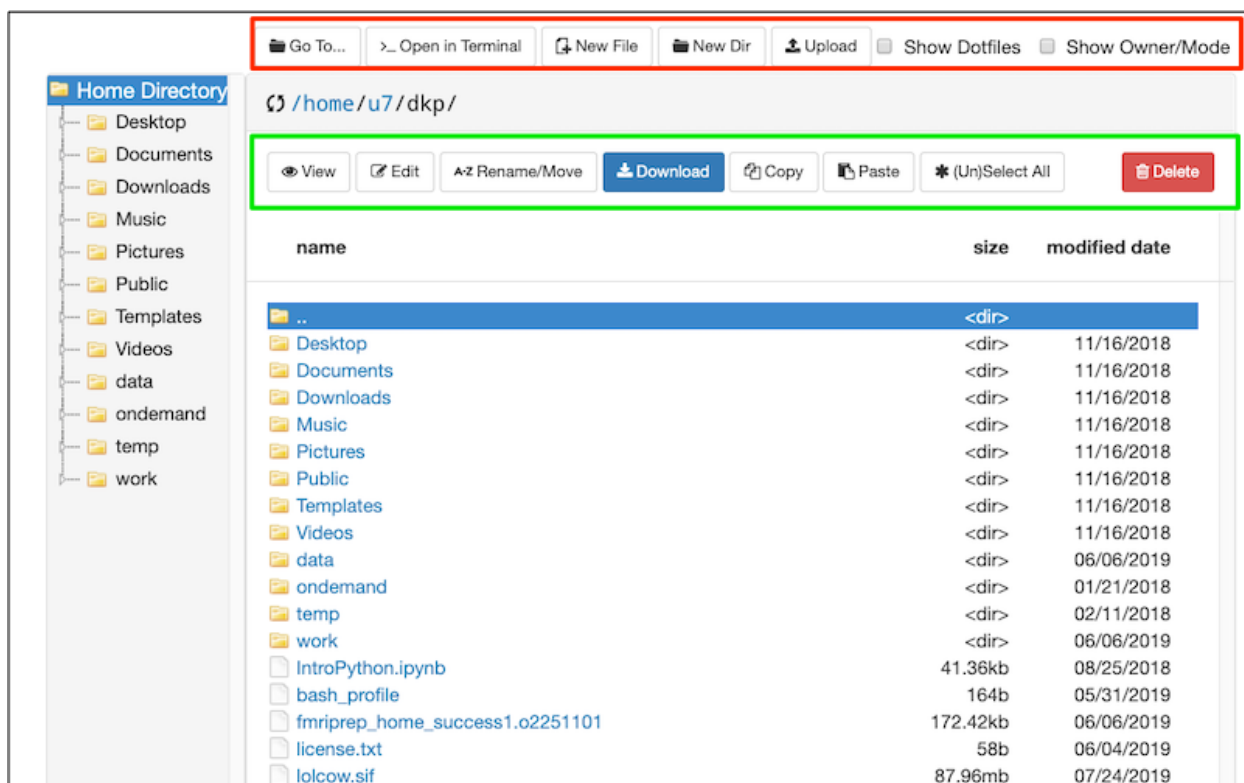
Fig. 2: Here the **OOD File Explorer** is displaying the contents of my home directory in the main panel. Regardless of what is displayed in the main panel, the **Home Directory** is always displayed on the left of the file explorer. In the **Red rectangle** (top) from left-to-right: I can use `Go To..` to type in the path I want. `>..Open in Terminal` provides the option to open a terminal window on one of the machines. Other buttons allow me to create new files or directories, and upload or download a file (something small like a text file or image). Checking the box `Show Dotfiles` will reveal hidden files and folders in the current directory. These hidden files and folders are usually important for configuration. Checking the box `Show Owner/Mode` will display permissions. Try checking each box to see what it does. In the **Green rectangle**: buttons allow various file operations, importantly the `Edit` button will open a text file, like a script, for editing. You can copy and paste directly from your local computer to the editor. Try various functions.

Note: You are NOT using your time allocation when you are browsing in OOD, or the File Explorer or the terminal, as long as you are not running anything intensive. Check out *Running out of time* to get a better sense of the differences between the login mode and the interactive or batch mode.

16.7.1 The File Explorer and Editor

Let's try using the File Explorer and the Editor. They are really nice and easy to use.

- On the top left of the File Explorer is a `Go To...` button. Click on it and type `/extra/dkp/Scripts`.
- Select `interact.sh`. Select `Copy`.

- On the left, the **File Explorer** shows your home directory. Click `Home Directory` at the top. Select `Paste`.

- You should now be in your home directory and have a copy of `interact.sh` with you.

- With `interact.sh` selected, click `Edit`.

- `interact.sh` is a bash script that can put you into interactive mode.

- Let's make a senseless edit so you can see how the editor works: Add the word `hello` to the COMMENT-BLOCK, like this:

```
-I # puts us in interactive mode
hello
-N interact # name the job interact
```

- The `Save` button on the upper left of the editor is activated by your change. Click it and save.

- There are some settings for how the editor appears on the right: font size, language (it should say `SH` indicating that this is a shell script), color scheme. Try something fun like changing the color scheme ;).

Note: You can use the *OOD* gui to create empty text files, and then use **Edit** to open one of these empty text files and paste in the contents of a script (e.g., interact.sh or buildSIF.sh) from the browser. Alternatively, you could download the scripts to your local machine and then upload them using OOD.

16.8 How the HPC differs from your Desktop Computer

Unlike the computer on your desk, the HPC is a group of computers with several different *host* machines: bastian, filexfer, login (login node for el gato), login2 and login3 (login nodes for Ocelote), and, of course, Ocelote and el gato (A new machine is being introduced in 2020). The storage and processing power are shared with other researchers. The different hosts provide different capabilities: For example, bastian handles security, the login nodes are for looking around and learning (that is where you are now), filexfer handles data transfers. Ocelote and el gato are two different computing clusters with different capabilities. Although you have more research time on Ocelote, the operating system is older and some things won't run (Ocelote's operating system is being upgraded to match el gato during 2020). See the section on *Host OS and Singularity* to understand why you might need to use el gato for some tasks until the Ocelote upgrade is complete.

16.8.1 Storage

No matter which host you are contacting, **your storage is the same**: Your home is the same. Your extra directory is the same.

Storage in your home directory is limited. Your HPC home directory will get backed up regularly BUT it is not very big. It certainly is not big enough to store your imaging data or even build most BIDS singularity containers.

You have more space in your `extra` directory, and it is probably enough for now. If you don't see an extra directory yet, be patient. It will be created. Later you can look into using xdisk to sign up for as much as 1 TB of space for up to 150 days. You can currently buy or rent additional space if you wish. Again, changes are coming to storage in 2020.

Check Disk Quota and Usage

Note: As of 4/06/2020: There are big changes afoot during Spring 2020: New SSD storage will be larger and faster. Storage space in your home directory will be increased dramatically to 50 GB, but will not be backed up. XDisk space up to 20 TB can be requested by PIs for 6 months, renewable once. /extra will be migrated. You won't be able to rent

---

space any more, but you can request space. The HPC group will work with you to make sure you have what you need. This is a good change. See the temporary Box documents HPC Storage Policies: and Data Migration Timeframe:.

16.8.2 Jobs: Interactive and Batch Mode

- Up to this point, you have been in login mode. You can look around in the directories and create small files, edit a script etc. In login mode you are not using any of your monthly research allocation hours.

- If you try to do a task on the login node that takes too many resources (e.g., run neuroimage processing or build a large singularity container), the task will be killed before it completes. Instead, you have to run such a task as a job. All jobs use your allocation hours and are tracked using the PBS job submission system:

    - **Interactive** A job can be submitted to use interactive time (e.g., any of the `Interactive apps` like a Desktop, Jupyter notebooks or R-studio available from OOD). You can even have interactive time at the terminal, which we explore below.

    - **Batch** A job can also be submitted to run in batch mode. A batch job is one where you submit a script to be run, and the cluster runs the job when the computing resources are available. We'll explore this too.

- Back on the OOD dashboard, click *Jobs Active Jobs*. It is probably not very interesting right now. You don't have any jobs running on el gato or Ocelote.

- Check out *Running out of time* to get a better sense of why you need to care about creating jobs. The example focuses on building and running singularity containers.

Running your First Job

Let's run a very small job so you can get a sense of how it works.

- Remember you copied a script called `interact.sh` to your home area? Open it in the editor if it is not still open.

- It is a bash script. This bash script starts `qsub` which defines the job. In the COMMENTBLOCK I describe some of the qsub flags.

- When we run this script, it'll run for up to one hour, after which it'll end. This is the `walltime=1:00:00`. (We'll delete the job sooner than that so as not to waste your allocation. You'll only be charged for the time you actually run).

- However, your alloction time is not just about walltime. One hour using 1 cpu and 4 GB of memory is different than one hour using 28 CPUs and 168 GB of memory. Because of this, your allocation time is more complicated than just walltime. Roughly it is the `# of CPUs * walltime`. And just to keep everyone honest, the amount of RAM is limited (e.g., 6 GB per CPU on Ocelote).

- Open an Ocelote shell, because you have LOTS of hours on Ocelote: You can open a shell from the OOD dashboard: `Clusters Ocelote Shell Access`. Or, in the File Explorer, select `>_Open in terminal Ocelote`.

- Ensure that interact.sh is executable:

```
[dkp@login3 dkp]$ chmod u+x interact.sh
```

- In the terminal window, type `./interact.sh` to see the usage message (the `./` is necessary if you have not got your home directory in your path):

```
[dkp@login3 dkp]$ ./interact.sh
```

- You see the usage message:

```
this script will put you in interactive mode with 4 cpus,
24 gb of ram and one hour of walltime
You must enter the group to be charged
e.g., interact.sh dkp
=============================
These are the groups you belong to
dkp current allocations:
-----------------------------
Group                    Type                    Available Time
-----                    ----                    --------------
dkp                      Standard                35999:39
dkp                      ElGato Standard         7000:00
```

The script will request a modest 4 cpus, 24 GB of RAM and 1 hour of walltime (walltime is just clock time). That should do the job for building a large *.sif file (but we are not going to do that right now). Here is what this process looks like when you run it with your group name:

```
[dkp@login3 dkp]$ ./interact.sh dkp
starting interactive session with 4 cpus, 24 gb of ram and 1 hour of wall time
qsub: waiting for job 2428450.head1.cm.cluster to start
qsub: job 2428450.head1.cm.cluster ready
[dkp@i0n1 ~]$
```

- The first prompt `[dkp@login3 dkp]$` indicates that I am in login mode.

- I request an interactive node for group dkp: `interact.sh dkp`.

- `interact.sh` tells me how many resources I will get: `starting interactive session with 4 cpus, 24 gb of ram and 1 hour of wall time`.

- At first, qsub reports it is waiting (3rd line above), so your interactive mode has NOT started yet. Give it some time (usually a few minutes).

- qsub will report when it is ready (4th line above), and then give you an interactive prompt, like this `[dkp@i0n1 ~]$`. Note the differences between this prompt and the login prompt.

- Congratulations! You are in interactive mode, not login mode!

Just because you start the interactive session with this request for 1 hour of walltime, does not mean you have to use the whole hour. You can delete the job. Back on the OOD dashboard, click *Jobs Active Jobs*. You should see your job listed:

Fig. 3: Click the disclosure triangle (in the green box) to see more details of the job. Once you have opened the job details, you'll see a red delete button on the lower right, so you can end the job.

You can also learn about the job in the shell window. For example, qsub told me the name of my job when it started: `2428450.head1.cm.cluster`. If you are not sure what your job is called, you can ask for your job names like this (use your own groupname instead of dkp):

```
qstat -wa -u dkp
```

Your jobs will be listed. Their status is also indicated: `Q` for queued; `R` for running. You can also look at the qsub log file as it is being created:

```
qpeek 2428450
```

In addition to being able to delete a job in the OOD dashboard, you can delete it in the terminal window, like this (you have to use your job name though):

```
qdel 2428450.head1.cm.cluster
```

Job deletion will take a moment. But then you will be charged only for the time interactive mode was actually running. This is a general principle, if you ask for more time and processing power than you use, you'll only be charged for what you use. However, if you ask for a LOT of resources, it may take longer for your job to actually start.

You can view your time allocations like this:

```
va
```

> **Warning:** The `interact.sh` script differs from most job submissions because all you have to do is type the script name. Usually, to run a job script you must type `qsub` before your script name. See *Running out of Time*.

Additional script examples are available on bitbucket and in /extra/dkp/Scripts.

Running Lots of Jobs

You are probably not interested in the HPC for running a single small job.

- **for loop**: Although you could write a for-loop to spawn a lot of jobs, or run job after job at the command prompt, this can overload the system scheduler and lead to problems with overall performance (see Best Practices).

- **BIDS** allows you to process multiple subjects at a time, but the processing is sequential, and if one subject crashes, none of the following subjects will get run. In addition, you need to calculate the required CPU, memory and walltime resources for the whole set of subjects.

- **qsub arrays** The most efficient way to run lots of jobs is with a job array. Unfortunately, job arrays work best on consecutively numbered jobs (e.g., 1,2,3,4) and we tend to want to run on subject directories, that probably are not consecutive.

- **qsubr** To address the problem of non-consecutive subject directories, see the script qsubr (There is also a copy of qsubr in /extra/dkp/Scripts on our UofA HPC).

qsubr will create a qsub array job for each subject in a list of subjects (one subject per row in a text file) you pass it:

```
qsubr Scripts/arraybip2prep.sh SubjectLists/plante333-334.subjects
```

For qsubr to process a script, you must make some additions to that script so that it knows it has to look at a subject list and iterate through rows:

```
##########################################################
#### BEGIN STUFF TO ADD TO ARRAY SCRIPT CALLED BY QSUBR ####

### Change to the directory where the PBS script was submitted. This is harmless.
cd $PBS_O_WORKDIR

### This is critical for any call to qsubr. It gets the subject names from the list.
### Note that the subject list can be specified, but defaults to subjects.txt in the␣
↪current directory if it is not specified.
### Pull filename from line number = PBS_ARRAY_INDEX
Subject="$( sed "${PBS_ARRAY_INDEX}q;d" "${SUBJS:-subjects.txt}")"

### The following is useful for documentation purposes.
### The array index and subject number get echoed to every output file produced by␣
↪qsub.
### Print job information to each output job
loginfo="JOBNAME=$PBS_JOBNAME, JOB ID: $PBS_JOBID, Array Index: ${PBS_ARRAY_INDEX},␣
↪Subject: sub-${Subject}"

### Also create a log file for the job that echos the subject number and index of␣
↪each subjob to a single log file.
echo ${loginfo} >>${PBS_JOBNAME}.log
echo ${loginfo}

#### END STUFF TO ADD TO ARRAY SCRIPT CALLED BY QSUBR ####
```

The variable `Subject` is then passed to the singularity run command in exactly the same way you passed Subject to qsub in the run scripts, e.g.:

```
singularity run --cleanenv --bind ${MRIS}:/data /extra/dkp/singularity-images/mriqc.
↪sif /data /data/derivatives/mriqc participant --participant_label ${Subject} -w $
↪{STUFF}/mriqc_work --verbose-reports
```

qsubr will create a qsub array job for each subject. It will treat each as a separate job, so all you need to know is what cpu, time and memory resources one subject job requires. Whenever there is room in the HPC job queue to run one or more of your jobs, they'll start. View working examples (anything called array*.sh) of these array scripts.

16.8.3 Transferring Files

You can use the command line or graphical tools to transfer data to and from the HPC. Your allocation time is NOT used for transferring files, however, if you try to transfer large files or lots of files on the login node, your transfer will be killed. Options are described in detail on the Transferring Files page.

Tiny Files

Small files can be moved using **Upload** and **Download** in the OOD file explorer. For example, this should work for uploading or downloading a text file or a single anatomical image. If you ask to transfer something that is too big, it'll just fail (but may not give a good message), so be suspicious and check.

Medium Sized Files

Here's an example of using `scp` from the commandline to transfer a large singularity container from my local mac to the HPC. It is also be a reasonable solution for a single subject BIDS dataset:

```
scp -v bids_data.zip dkp@filexfer.hpc.arizona.edu:/extra/dkp
```

I have to indicate who I am on the hpc: `dkp`. The transfer is handled by `filexfer.hpc.arizona.edu`. But never fear, the file will end up in my `/extra/dkp` directory as I specify. I have 200 GB of space in `/extra/dkp` and only 15 GB in my home directory, so I generally will NOT want to transfer imaging files to my home directory. This is a reasonable solution for data transfers of less than 100 GB, though not as fast as Globus.

Big Files and Directories: Globus

- Globus is the preferred tool for transferring data to and from the HPC, and even between directories on the HPC. There is a description of the Globus setup on the Transferring Files page.

- Globus provides a downloadable program for your local computer so you can treat your computer as an endpoint. You interact with the endpoints in a browser window with two panels (e.g., one panel for your local machine and one for the HPC).

> **Warning:** Check the preferences in your locally installed **Globus Connect Personal** to ensure that you have access to the directories you need (by default, you only have access to your home directory). In addition, you must explicitly tell Globus (Under its Preferences) to allow you to write to these directories.

> **Warning:** Globus does not transfer symbolic links. That is, if `ALink` points to `AFile`, then `Afile` will be transferred, but `Alink` will not. If you need symbolic links, you'll have to tar up your directory locally and then untar it on the HPC.

16.8.4 Optional Section: Return to a Previous Terminal Session Using Screen

On your personal mac or linux computer, each terminal is open in a particular directory. You can scroll back and see the commands you ran, and you can watch the output of a long process as it appears. If you log in to the HPC, you have this up until your terminal session ends (either you close it or it times out). When you log back in later, you are at a new fresh terminal that has lost this potentially valuable information. Fortunately, there is a unix commandline tool called `screen` that can help with this. Screen saves sessions so you can reconnect later, even though you may have closed your HPC session. Your process keeps running in the screen and you can check on how it is getting along by attaching to it. Here is a brief tutorial on getting started with screen.

Screen Tutorial

Let's see if you have any screens (probably not):

```
screen -list

No Sockets found in /var/run/screen/S-dkp.
```

Assuming you are NOT currently in a screen session (and you aren't if you've never done this before), you can **create a new screen session** like this:

```
screen
```

You will be switched to that new screen. You can **display the name of your current working screen session** like this:

```
echo $STY

1851.pts-4.login2
```

The screen has a 3-part name: `screenprocessID.sessionname.host` e.g., `27589.pts-4.login2`

After creating the session, you should be able to refer to it with its screenprocessID or sessionname.hostname or both: (`1851.pts-4.login2` or `1851` or `pts-4.login2`).

Detach from a screen Let's say you don't want to be in the screen anymore, but you want to be able to return to it later. You should detach from the screen:

```
screen -d 1851
```

Note: If you have only one attached screen, then you can simply type `screen -d`, but as soon as you have multiple attached screens, you need to be specific.

Now look at the state of things:

```
screen -list

There is a screen on:
      1851.pts-4.login2       (Detached)
      1 Socket in /var/run/screen/S-dkp.

echo $STY
```

Your screen exists, but is detached. `echo $STY` returns nothing because you are no longer in a screen session.

You can also create a **screen with a meaningful sessionname**, and you will be switched into it. In this case, the name is 2 parts: screenprocessID.sessionname:

```
screen -S fred

echo $STY

4032.fred
```

You can list your screens from inside a screen session. If we **list the screens from inside fred**, we see that fred is attached:

```
screen -list

There are screens on:
      1851.pts-4.login2  (Detached)
      4032.fred          (Attached)
```

The fred screen can be referred to as `4032.fred` or `4032` or `fred`. Let's **detach from fred**, and then check that we are not in a screen session with `echo $STY`:

```
screen -d fred
echo $STY
```

(continues on next page)

```
There are screens on:
      1851.pts-4.login2  (Detached)
      4032.fred          (Detached)
```

Both screens are now detached.

Re-attach to a screen session:

```
screen -r fred
echo $STY

4032.fred
```

From fred, create a third screen, joe. We should be switched to joe and both joe and fred should be attached. Check with `echo $STY` and list the screens:

```
screen -S joe
echo $STY
screen -list
There are screens on:
      27376.joe          (Attached)
      1851.pts-4.login2  (Detached)
      4032.fred          (Attached)
3 Sockets in /var/run/screen/S-dkp.
```

Use `-x` instead of `-r` to switch to the attached screen:

```
screen -x joe
```

> **Warning:** If you create screens within other screens (as we created joe from within fred) you can get into trouble. Ctrl AD can get you out, as you'll probably do this at some point. However, it is best to detach from one screen before creating a new one.

When you have lost your prompt! From within a screen session that is busy or messed up, you can press Ctrl AD (Hold control, and press A and then D; these are not uppercase).

Once you have started a long process, like running a neuroimaging container, you can use Ctrl AD to detach from the screen (or you can just close the HPC browser tab and your screens will be preserved).

Kill Perhaps you have created a lot of screens. You'd like to get rid of these screen sessions altogether (not just detach from them). Here's my current list:

```
screen -list

here are screens on:
      1851.pts-4.login2  (Detached)
      4032.fred          (Attached)
      27376.joe          (Attached)

3 Sockets in /var/run/screen/S-dkp.
```

> **Note:** Your situation may be different depending on how much you've mucked about creating and detaching. But you probably need some cleanup.

Here we kill a screen and then list the screens to ensure that it is gone:

```
screen -X -S 1851 kill
[screen is terminating]

screen -list
There is a screen on:
        4032.fred       (Attached)
        27376.joe       (Attached)
2 Socket in /var/run/screen/S-dkp.
```

Note that 1851 is no longer listed after being killed. Both detached and attached screens can be killed. In additon, when you are in a screen session, you can `exit` to both detach and kill the session:

```
screen -r fred
exit
[screen is terminating]
screen -list
  There is a screen on:
          27376.joe       (Attached)
  1 Socket in /var/run/screen/S-dkp.
```

Note: When you terminate a screen, the prompt may jump to the bottom of the terminal window!

There is more, but this should get you started. BTW, the screen utility may be lurking on other machines you use, like your mac.

16.8.5 Deidentifying Data

Remember, the HPC storage is **not HIPAA compliant**. Your **data must be deidentified** before being uploaded. This means you should convert it to NIFTI and then deface any T1 images. The most commonly used defacing tool is the Freesurfer deface tool. Here's a simple script, deface.sh to call the deface tool. The deface script assumes you have the following three files in your path:

- a binary of mri_deface for your computer (and the old mac binary is still works on Mojave),

- talairach_mixed_with_skull.gca

- face.gca

Although defacing is a pain, it will make your data suitable for upload to data sharing sites if you ever need to do that.

Next: Learn about Singularity Containers

16.8.6 Optional Section: SSH TO HPC

Feel free to skip this section if you are not especially interested right now. If you prefer the command line, you can ssh to the HPC e.g.,:

```
ssh dkp@hpc.arizona.edu
```

Of course, you'll need to put in your own username, which probably is not `dkp`, And then you'll go through the authentication process:

```
Password:
Duo two-factor login for dkp
Enter a passcode or select one of the following options:
Duo Push to XXX-XXX-XXXX
Phone call to XXX-XXX-XXXX
SMS passcodes to XXX-XXX-XXXX (next code starts with: 2)
Passcode or option (1-3): 1
Success. Logging you in...
Last login: Sun Jan 21 19:16:14 2018 from http://66-208-204-2-pima.hfc.
→comcastbusiness.net
***
     The University of Arizona HPC systems will be down from
      2019-07-31 07:00:00 thru 2019-07-31 18:00:00
        for Quarterly Software Updates; ALL HPC services will be unavailable.
***
===============
http://HPC.ARIZONA.EDU
===============
Please select a target system to connect to:
(1) Ocelote
(2) El Gato
(Q) Quit
(D) Disable menu
```

To log in to el gato, you'd type `2` and see something like this:

```
Sending you to El Gato...
The authenticity of host 'http://login.elgato.hpc.arizona.edu (10.140.86.3)' can't be␣
→established.
RSA key fingerprint is XXXXXXXXXX.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'http://login.elgato.hpc.arizona.edu,10.140.86.3' (RSA) to␣
→the list of known hosts.
[dkp@login ~]$
```

Singularity on the HPC

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2019_07_23
Date Updated: 2020_04_06
Tags: UNIX, containers, bids
OS: UNIX

17.1 Introduction

The goal of this page is to describe running and building containerized BIDS workflows on the HPC. Containerized technologies are changing fast, so check the *Date Updated* above to ensure you are looking at reasonably current documentation.

For security reasons:

- Docker cannot be used on the HPC, but Docker containers can be converted to Singularity containers which can be run on the HPC.

- It is also not possible to build a Singularity container from a Singularity recipe file on the HPC, but we can build a Singularity container by pulling an existing Docker container such as a BIDS container.

For details of working with BIDS containers available to you on our HPC, see the page *Bids Containers*.

17.2 Build Singularity Containers from Dockerhub

To get a basic idea of how singularity containers work, you should try this. You do not even have to be in interactive mode. You can build a Singularity *.sif container on the HPC from an existing Docker container posted on dockerhub:

```
module load singularity
module list
```

```
module help
singularity build lolcow.sif docker://godlovedc/lolcow
```

Above, we run four commands:

- The first command, `module load singularity`, makes singularity tools available to us on the HPC by loading the associated module.

- The second command `module list` lists what modules we have loaded. This is not strictly necessary, but it will tell you which version of singularity has been loaded.

- The third command `module help` is also not necessary, but tells you more about the module command and how to use it.

- The fourth command `singularity build lolcow.sif docker://godlovedc/lolcow` builds a singularity file `lolcow.sif` (call it whatever you want), by pulling the associated Docker container from dockerhub.

- Once the container has built, you can run the Singularity container thus:

  ```
  singularity run lolcow.sif
  ```

- You should see a clever saying produced by an ascii cow, something different every time you run it.

17.3 Build Singularity Containers from Recipes

- If you have a Singularity recipe file, you **cannot** build it on the HPC.

- You can build a Singularity container from a recipe on Sylabs Cloud. I believe there are some size and time limits, but I have successfully built the 1.6 GB BIP2 container in 7 minutes on Sylabs Cloud.

- If your recipe is too resource-intensive to build on Sylabs cloud, you can use **Cyverse Atmosphere** if you sign up for an account. See *Singularity on Cyverse*.

- You could build a linux machine and install Singularity (not for the faint of heart).

17.4 Running a BIDS Singularity container

- Running a job like fMRIPrep or mrtrix3_connectome will likely take overnight (depending on your options). It is better to write a batch job specification that overestimates what you need to finish running a subject than to underestimate.

- If you underestimate, the job will be killed before it is finished (so, 3 hours is not enough, 2 CPUs might not be enough).

- If you overestimate, you'll be charged only for the time you actually use (so, maybe try 48 hours?).

- If you vastly overestimate (e.g 200 hours) then your job may not start until the HPC has that much time available to allocate to you. So, it may take some experimentation to find the right balance.

- If you have run the Docker container on your local machine, that will give you a rough estimate. You could double that number of hours and see how it goes.

- See *BIDS containers* for details about individual BIDS containers available on the U of A HPC system and the corresponding run scripts and resource estimates.

17.5 Host OS and Singularity Interactions

For the most part, what is inside your container is separate from what is outside your container. That's the point of containers. However, there can be interactions. Consider the following case:

MRtrix3_connectome is built on Ubuntu 18.04. You can see this by looking at the first line in the dockerfile.

Ocelote is (at the time of this writing) on CentOS release 6.10:

```
[dkp@login2 dkp]$ cat /etc/issue
CentOS release 6.10 (Final)

[dkp@login2 dkp]$ uname -a
Linux login2 2.6.32-754.12.1.el6.x86_64  #1  SMP Tue Apr 9 14:52:26 UTC 2019 x86_64␣
→x86_64 x86_64 GNU/Linux

[dkp@login2 dkp]$ cat /etc/*elease
CentOS release 6.10 (Final)
Cluster Manager v8.1
slave
LSB_VERSION=base-4.0-amd64:base-4.0-noarch:core-4.0-amd64:core-4.0-noarch:graphics-4.
→0-amd64:graphics-4.0-noarch:printing-4.0-amd64:printing-4.0-noarch
CentOS release 6.10 (Final)
CentOS release 6.10 (Final)
```

If I try to run even the simplest command calling the mrtrix3_connectome.sif container from the ocelote shell prompt, I get this message:

```
[dkp@login2 singularity-images]$ singularity run mrtrix3_connectome.sif -v
FATAL: kernel too old
```

If I log in to el gato, several things are different. First `cat /etc/issue` isn't helpful on el gato:

```
[dkp@login dkp]$ cat /etc/issue
\S
Kernel \r on an \m
```

The other two commands work fine on el gato:

```
[dkp@login dkp]$ uname -a
Linux login 3.10.0-957.10.1.el7.x86_64 #1 SMP Mon Mar 18 15:06:45 UTC 2019 x86_64 x86_␣
→64 x86_64 GNU/Linux

[dkp@login dkp]$ cat /etc/*elease
CentOS Linux release 7.6.1810 (Core)
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
```

(continues on next page)

```
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"

CentOS Linux release 7.6.1810 (Core)
CentOS Linux release 7.6.1810 (Core)
```

As you can see, the operating system is much newer on **el gato** (`CentOS Linux release 7.6.1810 (Core)`) than on **Ocelote** (`CentOS release 6.10 (Final)`). It turns out that el gato is new enough to handle a container built on Ubuntu 18.04. When I ask for the version of the mrtrix3_connectome.sif from el gato, I get a reasonable answer:

```
[dkp@login singularity-images]$ singularity run mrtrix3_connectome.sif -v
BIDS-App 'MRtrix3_connectome' version 0.4.2
```

Note: As of 2/14/2020: Some nodes on Ocelote are available with CentOS7. See the Including CentOS7 documentaion. To use these nodes, you must add `os7=True` to your qsub select statement, e.g., `select=1:ncpus=28:mem=12gb:np100s=1:os7=True`

17.6 Building Large Containers

The process for building *lolcow.sif* works fine for a small container, but for large containers, you run into two problems: **Running out of Space** and **Running out of Time**.

Note: You do not have to build containers yourself if you are happy with the ones I have built. Look in `/extra/dkp/singularity-images`. The permissions should be such that you can run any container I have in here.

17.6.1 Running Out of Space

On our University of Arizona HPC and on Cyverse virtual machines (Atmosphere), even if you have access to many GB of space, the space IN YOUR HOME DIRECTORY is likely to be very small. When Singularity builds a *.sif file, as illustrated for *lolcow.sif*, it pulls tarballs from Dockerhub into the hidden `.singularity/docker` and `.singularity/cache` subdirectories in your home directory. Singularity quickly runs out of available space in your home directory and then complains and dies. Below I describe how to circumvent this problem:

- Move `.singularity/docker` and `.singularity/cache` FROM your home directory TO a directory with more space (e.g., `/extra/dkp` in my case)

- Add information to your shell configuration file to tell Singularity where to find these important directories. I added the lines below to my `.bash_profile`:

```
export SINGULARITY_PULLFOLDER=/extra/dkp/build_sif/docker
export SINGULARITY_CACHEDIR=/extra/dkp/build_sif/cache
```

You should **modify these lines for your own directory** (not `dkp` please). Now you can submit a batch job to build the Singularity container, for example you could modify buildSIF.sh for your own group and chosen directory.

Note: As of 2/14/2020: We are expecting to have 50 GB of space in home sometime during the Spring of 2020 as our new cluster and enhanced storage comes online. You ought to be able to pull large Docker containers and build

them into Singularity containers with this much space (unless you fill up the space). Other Singularity-related tools for building and storage may be coming our way, so stay vigilant.

17.6.2 Running Out of Time

As described in *Batch Jobs and Interactive Mode* when you log on to the HPC, you are in login mode. In login mode, you are not using any of your allocated monthly time allowance but you can only do small tasks (like building *lolcow.sif*). The HPC will kill your job if you try to build a large Singularity container (like any of the BIDS containers). One solution is to submit a batch job with a script like the buildSIF.sh (this builds the fMRIPrep container). Batch jobs are special scripts submitted with `qsub`. These scripts combine instructions to the PBS batch system and bash commands. Submit the script with qsub, like this:

```
qsub buildSIF.sh
```

If you want to actually work at the commandline (like you did with the lolcow build), then you have to *submit a batch job that puts you into interactive mode at the command line*.

17.7 BIDS Containers

Cyverse for Neuroimaging

Maintainer: Dianne Patterson Ph.D. dkp @ email.arizona.edu
Date Created: 2019_07_24
Date Updated: 2019_08_10
Tags: UNIX

This page provides information for the neuroimaging community about getting started with Cyverse at the University of Arizona. The focus is on a Cyverse service called Atmosphere which allows you to build or use existing virtual unix machines. An advantage of the Cyverse virtual machines over the HPC is that you have administraive access. This means you can build Singularity containers from Singularity definition files (which you cannot do on the HPC). You can also run Docker on a Cyverse virtual machine (again, because you have administrative access. There are disadvantages as well: the HPC provides access to GPU processing, Cyverse does not provide that (yet). Look at the **Date Updated** above to ensure that you are looking at reasonably current documentation. You can interact with Cyverse Atmosphere virtual machines at the command line or through a graphical user interface, but to use these resources effectively you should probably have some basic knowledge of *unix*.

Sign up for Cyverse, and then sign up for Atmosphere (a service within Cyverse that provides virtual machines).
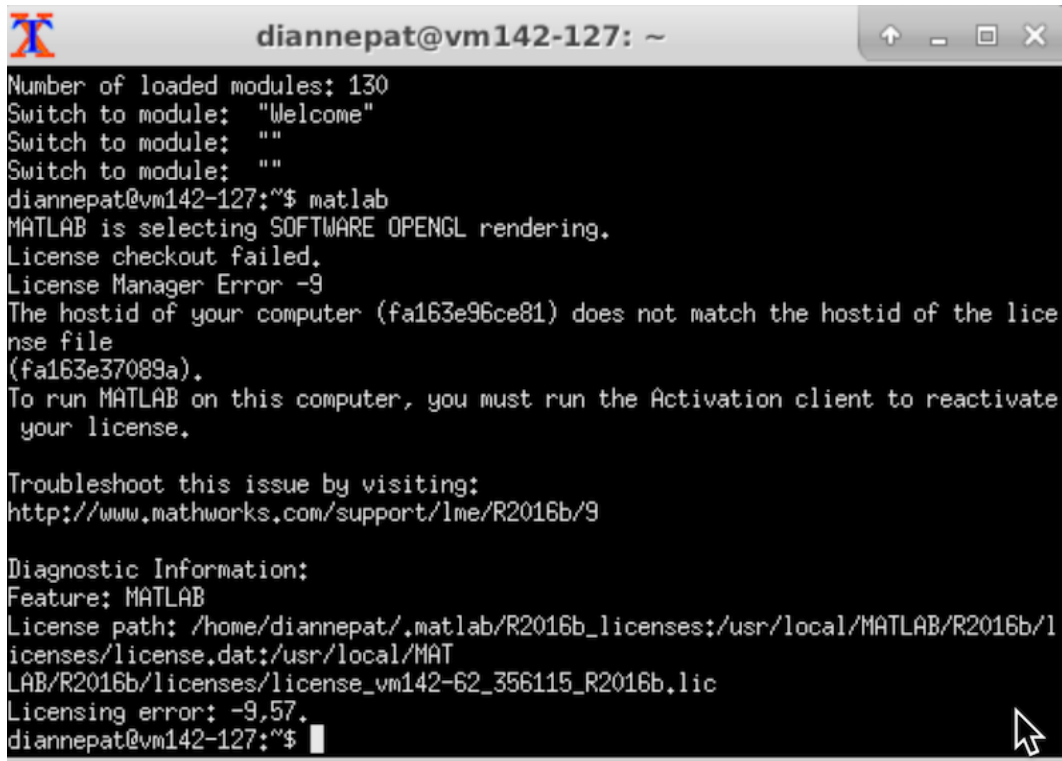
18.1 Neuroimaging VM, version 1.1

This is an Ubuntu 16.04 machine with a graphical user interface (GUI), Neurodebian and Matlab. Core tools for analysis of fMRI, DWI and related images have been installed. In brief, this includes AFNI, FSL, Freesurfer, Slicer3D, ANTS, itksnap, Convert3d, MRIconvert, Connectome Workbench and Connectome viewer. In addition, Matlab2016b is installed and a number of Matlab toolboxes for neuroimaging: SPM8, SPM12, GIFT; For network analysis: BCT, Conn, DPABI; for EEG and MEG: EEGLAB and Brainstorm3. You can activate an instance with up to 16 cpus and 128 GB of ram, however, you Cyverse prefers that you request fewer cpus and less RAM (and you are more likely to be able to start a machine with more modest requirements.

The following Matlab toolboxes are installed: Curve Fitting Toolbox DSP System Toolbox Global Optimization Toolbox Image Processing Toolbox Neural Network Toolbox Optimization Toolbox Signal Processing Toolbox Statistical and Machine Learning Toolbox Wavelet Toolbox

18.1.1 Activating Matlab on Neuroimaging VM

If you need to use matlab on the neuroimaging VM, you must activate it with your own credentials (because the hostid of the VM is not consistent). Start the activation script from the desktop or from an Xwindows capable system. The GUI interface pops up, you log in to your Matlab account (if you don't have one, you can sign up if you are a member of the UofA community).

Run the activation script:

```
ActivateMatlab
```

Follow the instructions. Various neuroimaging Matlab toolboxes are installed under /MATLAB_TOOLS. spm12 is in the path by default, but you will need to manually add any other tools to the path. Why? Because these tools can conflict, so you only want to put what you need in the path. For example, the installed version of Gift only works with spm8 (not spm12). But if spm8 and spm12 are in the path at the same time, there will likely be conflicts. fslview is installed (but not FSLeyes) mri_deface is installed (through neurodebian). This tool allows you to remove the face from structural images so that they do not violate HIPAA standards. You can call it with the appropriate talairach and face gca files by typing:

```
deface.sh
```

You can invoke the FSL command line tool slicer like this:

```
slicer (invokes a commandline fsl tool)
```

And you can invoke the Slicer3D gui:

```
Slicer
```

18.2 General How To and Gotchas

18.2.1 Root

You have sudo capabilities on your instance. You can do anything to it you want. HIPAA Cyverse data storage is not currently HIPAA compliant. See deface.sh above so that your files can be deidentified.

18.2.2 Bye Bye Home Directory and Modifications

Unlike other linux machines you may have used, as soon as you delete your instance, your home directory is gone with it. In fact, anything you have created on the instance is gone UNLESS you request that it be saved as a new revised image.

18.2.3 Build Your Own

#. Modify Neuroimaging and request an image of it. You must be running the instance when you request it be imaged, and LEAVE IT RUNNING until you receive notification that your revision is complete. #. Or, you probably want to start with u1604 desktop base 50gb root not featured. This is a private VM, so you'll need to ask for access to it. Other Cyverse base images only have 20 GB of install space for software. Given that most of our programs are 4-5 GB apiece, it is easy to run out of space installing neuroimaging software.

18.2.4 Web Desktop Tweaks

If you use the web desktop, I suggest you choose the "default panel" rather than the "empty panel". The default pane provides icons on the desktop for accessing the terminal window and several installed apps:

```
Desktop resolution
Display the available resolutions
>xrandr -q
1   800x600        30.00     60.00

2   1024x768       30.00     60.00

3   1024x576       30.00

4   1280x960       30.00     60.00

5  1344x1008       30.00     60.00

6  1344x756        30.00

7 1600x1200       30.00     60.00
```
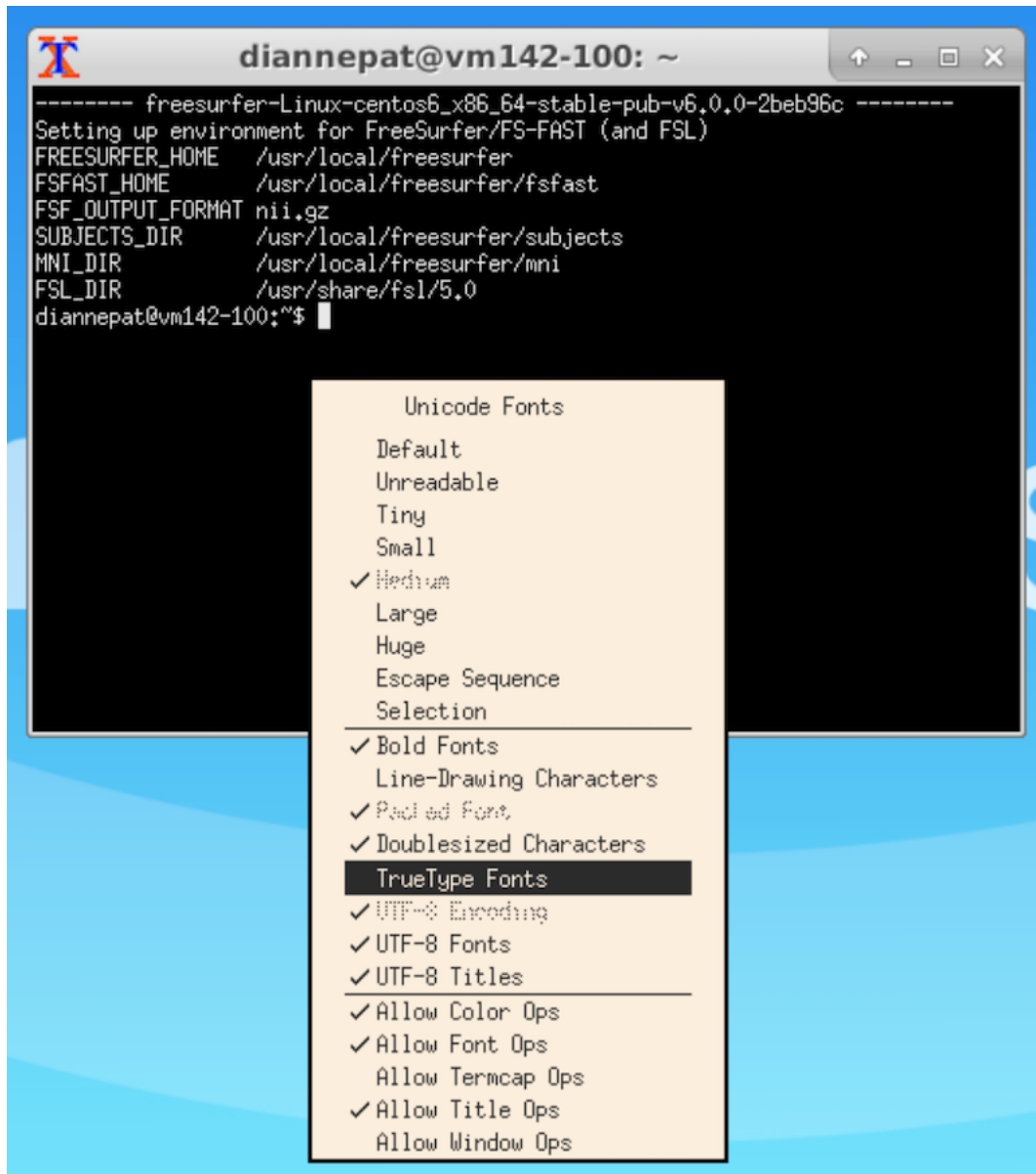
Invoke your preferred resolution (here 1600x1200):

```
xrandr -s 7
```

18.2.5 Terminal font size (on the desktop)

Select the terminal window and hold the Control button on your keyboard while right-clicking.

Choose TrueType Fonts. If that doesn't do it for you, control-right-click again and select the font size you want.

18.2.6 Data Transfer

Cyverse has two different places to put data: #. Datastore (which you can view from the Discovery Environment). Data can be transferred to and from the datastore without running a virtual machine. Cyberduck can be set up to transfer files. icommands is a more robust (linux and mac) command line transfer utility. #. Volume. See detail below about mounting and unmounting the volume from a VM. To move data to a volume, you must first run a VM and attach the volume to the VM. You could also transfer data directly to the VM instead of a volume, but there is no speed advantage. A volume is useful in that it persists when the VM is deleted. So, you could start a small VM, transfer your

data and tools to a mounted volume... in general, make sure the environment is all set up; then unmount the volume, delete the VM and start a big VM where you will actually run your processing.

- Local computer to Data store: Cyberduck

- icommands setup

- icommands usage

Ensure the following is in .bashrc:

```
# Added by icommands 4.1.9 CyVerse Installer on
# On Tue Sep 19 12:35:04 MST 2017
export PATH="/Applications/icommands/:$PATH"
# Adding plugin path for older Mac OS X systems
export IRODS_PLUGINS_HOME=/Applications/icommands/plugins/
source /Applications/icommands/i-commands-auto.bash
```

Initialize icommands:

```
iinit
```

If this is the first time, then you need to enter:

```
host: data.iplantcollaborative.org
Port: 1247
User: (your Cyverse username)
Zone: iplant
Password: (your Cyverse password)
```

Now you have have access to some simple linux commands as icommands. For example, assuming you are coming from a Mac or linux machine, `ls` will still give you a listing of your local machine files. However, `ils` will give you a listing of the datastore files.

Pay attention to what directory you are in on the datastore when you push or pull. Here are some example commands:

```
iput -h # provides a man page for iput
```

Here are some examples of moving data between my Mac (local computer), the Cyverse datastore and the running Neuroimaging virtual machine on Atmosphere. Sync local computer to datastore (at bash prompt on local computer):

```
iput /Volumes/Main/Exps/Data/ROTMAN/PPA -r -K ROTMAN
```

-K insures error correction by doing a checksum and creating a checksum file, which will speed up data transfer with irsync later (see below). The final **/** insures that PPA directory is copied into ROTMAN, not just the contents of the PPA directory.

Sync contents of subject dir on local computer to same subject folder on datastore:

```
irsync -r sub-5167 i:/iplant/home/diannepat/ROTMAN/YC/derivatives/sub-5167
```

Sync datastore to local computer (again, at bash prompt on local computer):

```
irsync -r i:/iplant/home/diannepat/ROTMAN/PPA /Volumes/Main/Exps/Data/ROTMAN/PPA
```

Sync datastore to VM (from bash prompt on VM):

```
irsync -r i:/iplant/home/diannepat/ROTMAN/PPA /vol_c
```

To get out of icommands entirely:

```
iexit full
```

In addition, on every Cyverse VM there is a script called cyverse_backup.sh which tries to make this copying easier. From the VM, type:
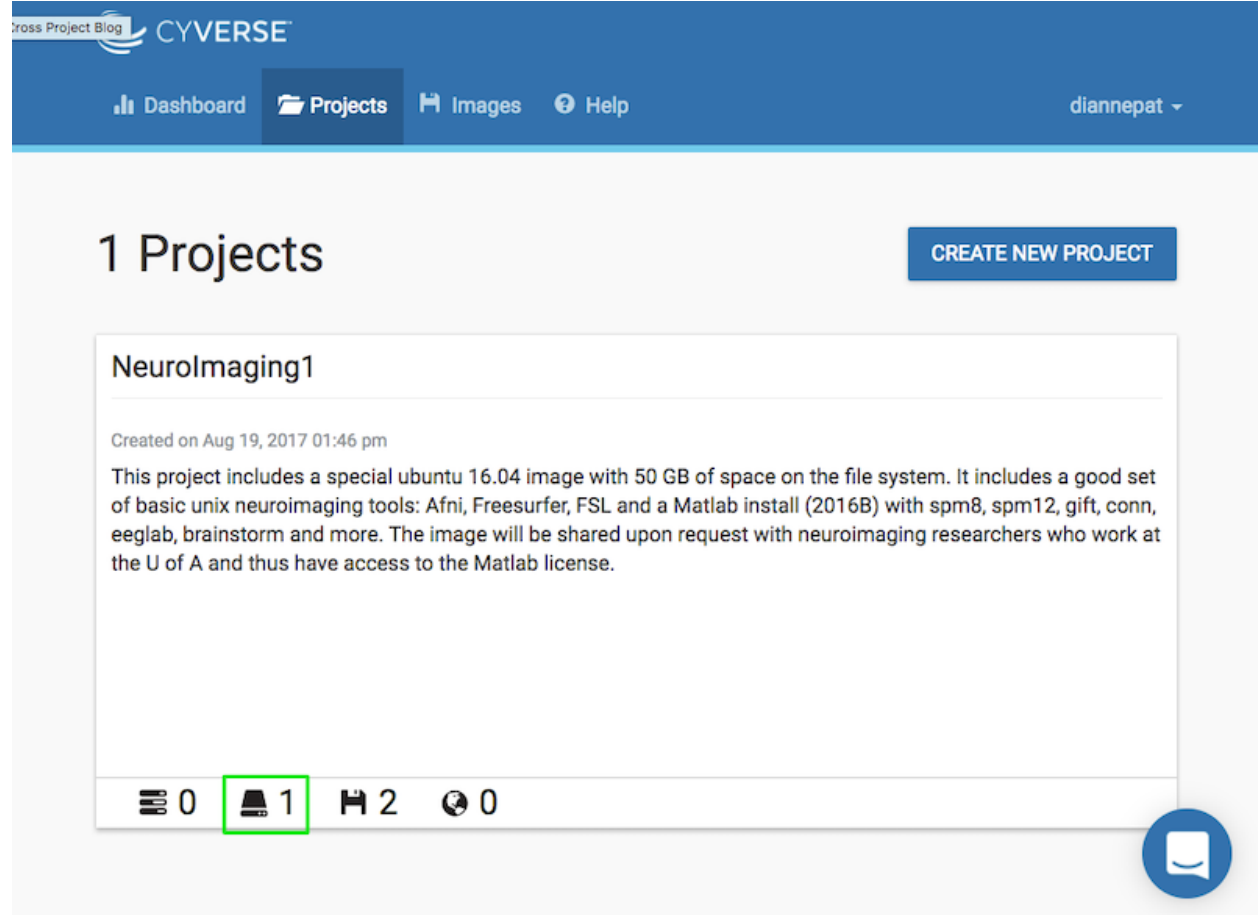
```
cyverse_backup.sh
```

At the prompt choose restore to copy TO your virtual machine or volume FROM the data store, or choose backup to copy TO your data store FROM your virtual machine or volume. See details here cyverse_backup.sh.
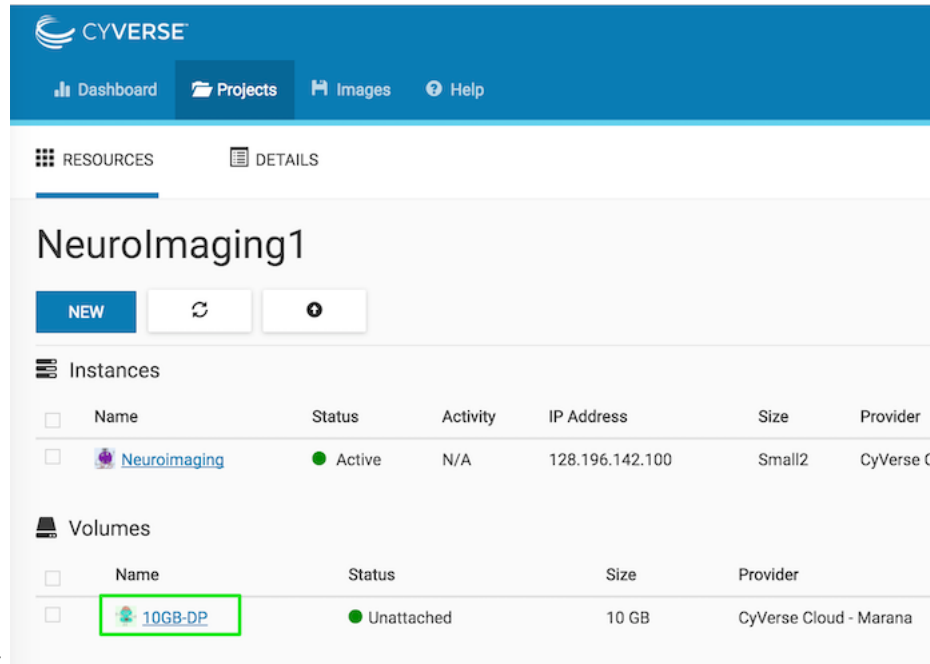
18.2.7 Mounting a Volume

See Attaching and Detaching Volumes.

From the Atmosphere web application, choose your project and click on the volume:

You should see your volume listed as unattached:
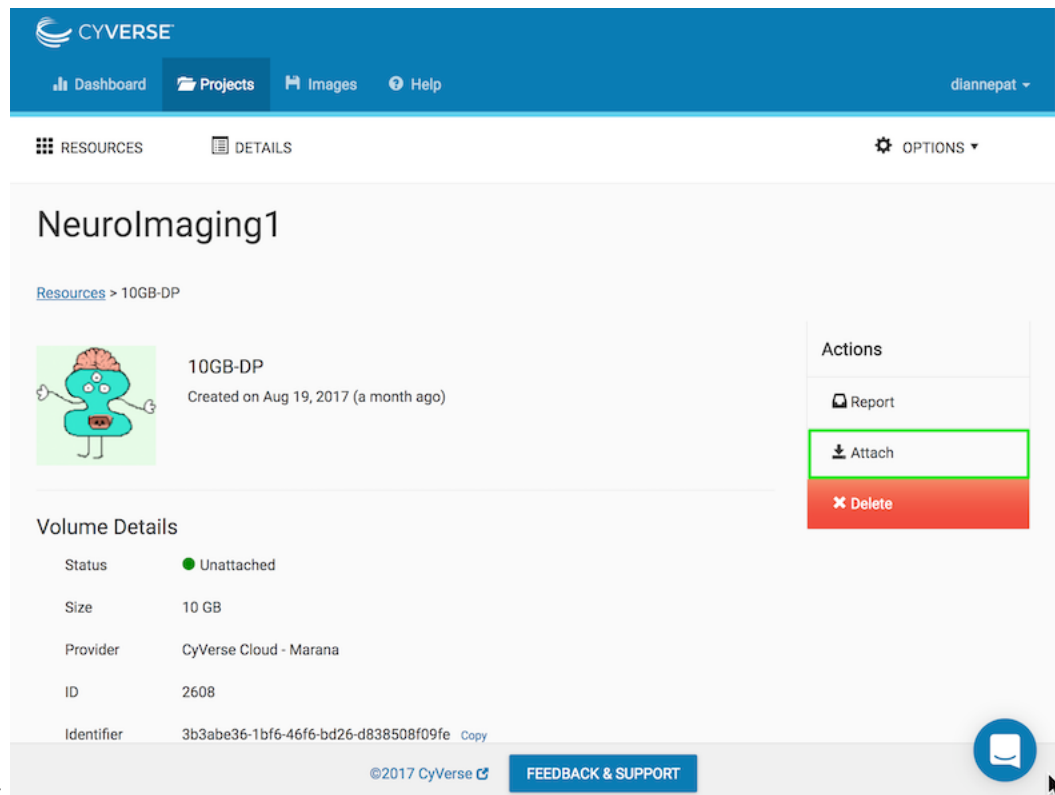
Click the attach button on the right:

After a few moments, the volume is attached and mounted. It will appear at the root level of your running virtual machine as vol_b or vol_c

Show you the mounted filesystems:

```
df -h
```

If the mounted volume is owned by root, or you do not have permission to write to it, you need to change the permis-

sions:

```
cd /
```

To change permissions:

```
sudo chmod a+rwx vol_c
```

To change ownership:

```
sudo chown myusername vol_c
```

Now you can use cyverse_backup.sh

18.2.8 Detach Volume before deleting the Instance

If the volume won't detach, ensure there are no desktops, shells or web shells displaying information in the volume directory. Check for any processes running on that volume with lsof:

```
lsof | grep /vol_c

atom       24638        diannepat   cwd        DIR             253,32      4096      ␣
→525174 /vol_c/bip/PROFILES

atom       24789        diannepat   cwd        DIR             253,32      4096      ␣
→525174 /vol_c/bip/PROFILES
```

End or kill those processes with kill -9:

```
kill -9 24638

diannepat@vm142-119:/$ lsof |grep /vol_c

atom       24789        diannepat   cwd        DIR             253,32      4096      ␣
→525174 /vol_c/bip/PROFILES
```

18.2.9 Singularity on Cyverse

If you would like to build a Singularity container from a singularity definition file, but you don't have a linux machine with root permission, then use a Cyverse VM. At the shell prompt type:

```
ezs
```

It'll ask for your password and install Singularity. See *Singularity on the HPC* to learn about how to use Singularity containers. See *the BIDS page* to learn about why you would use containers (Singularity or Docker).

If you are trying to build a large Singularity container and run into problems with space, then see *Running out of Space*.

CHAPTER 19

Indices and tables

- genindex
- search

Index

A

Affine transforms, **47**
applytopup, **47**
ASL, **47**

B

BBR, **48**

D

dcm2niix, **48**
Difference in Echo Times, **48**
Dwell Time, **48**

E

Echo Spacing, **48**
Effective Echo Spacing, **48**

F

Field Maps, **48**
FreeSurfer, **48**
FSL, **48**
FSLeyes, **48**

G

GIFT, **48**
grappa, **48**

N

NIFTI Headers and Orientation, **49**

P

Phase Encode Direction, **49**

Q

Q-form, **49**
Q-space Sampling, **49**

R

Reverse Phase Encode, **49**

S

S-form, **50**

T

Topup, **50**
Total Readout Time, **50**

W

Wavelets, **50**