fMRIprep Tutorial

Please ask questions!

This presentation is here to make your life easier if you want to use fMRIprep. I had to really bang my head against the wall for a while to get everything working and it’s likely that something that I now think is common knowledge is not. The only way I’ll know is if you ask!

Today's Structure

- Overview of what fMRIprep does
- Installing/running fMRIprep on a server*
- fMRIprep outputs
- Loading fMRIprep into CONN for further analysis

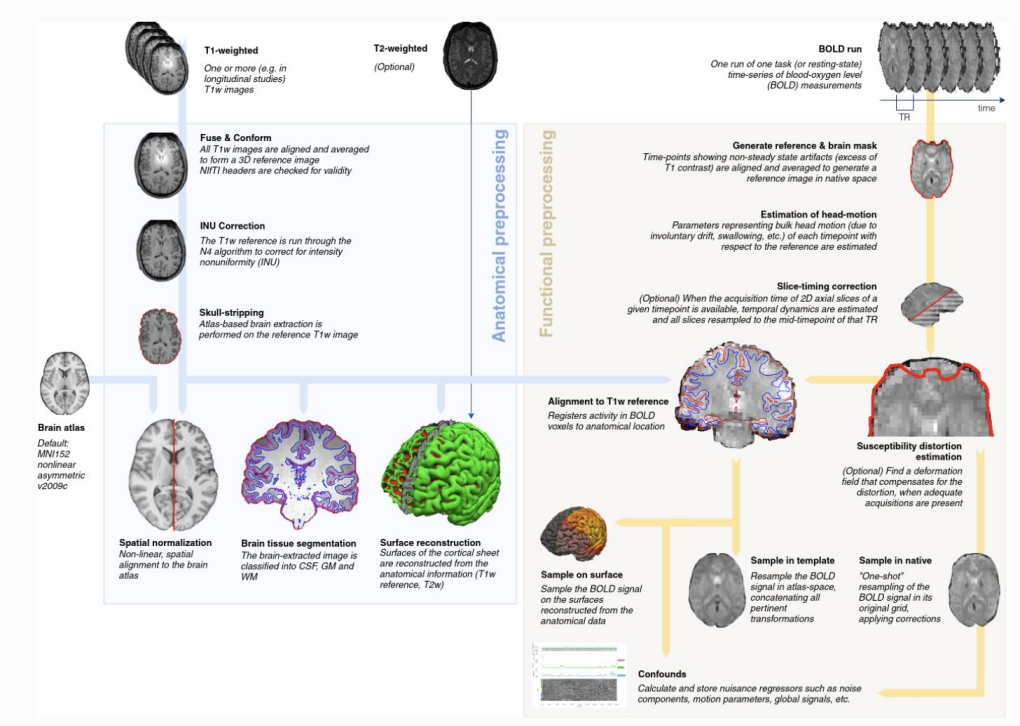
Most of the figures and code snippets can be found on https://fmriprep.org/en/stable/

* Everything I'll show uses a slurm based server. My understanding is that USC uses a PBS server. There is a direct analog for everything, you'll just have to look it up.

fMRIprep Overview

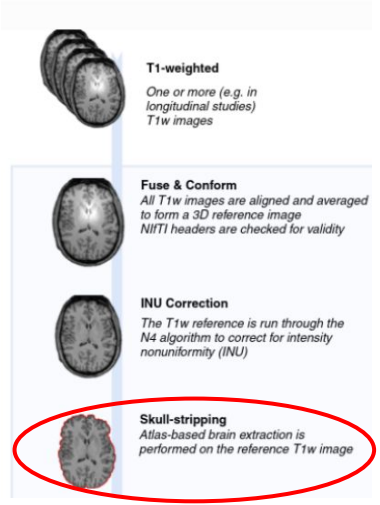
fMRIprep is an automated preprocessing pipeline that combines a variety of tools in one place to make it easy to run a state of the art minimal preprocessing pipeline on your data.

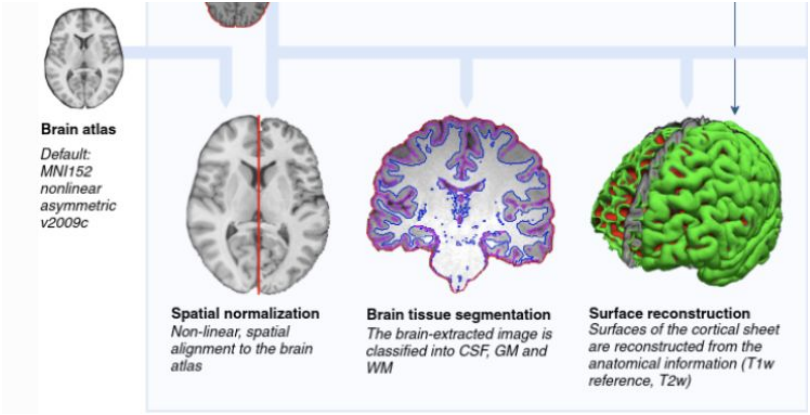
fMRIprep uses FSL, ANTs, Freesurfer, AFNI.



Overview of the pipeline: we'll break this down in a minute

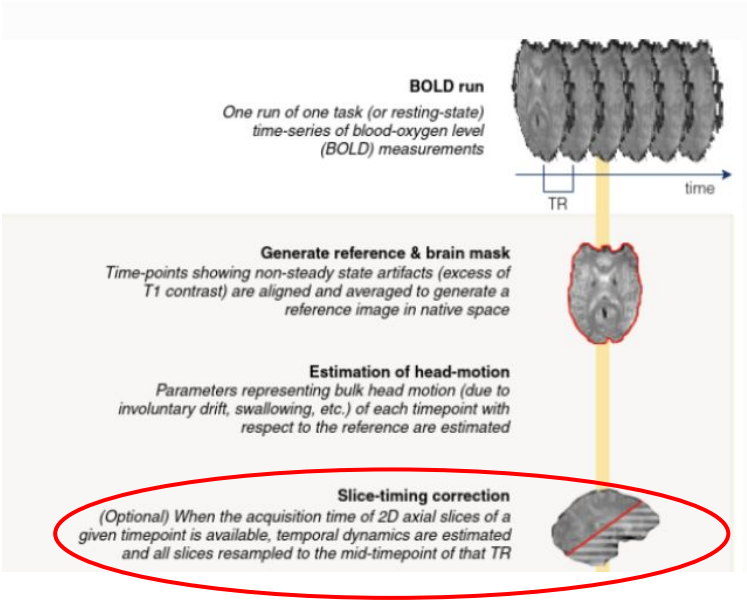
Structural Preprocessing

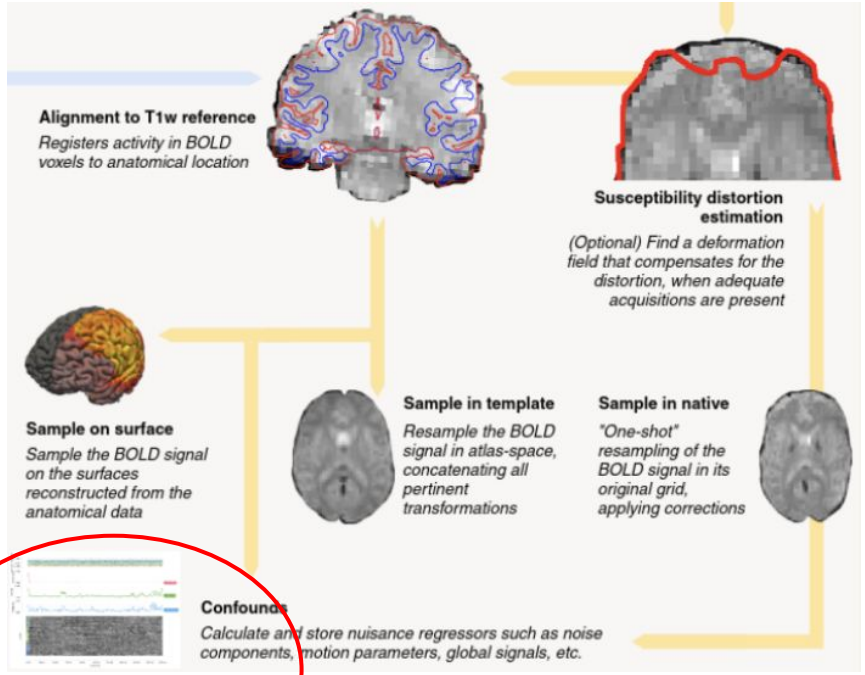




Full Details here: https://fmriprep.org/en/latest/workflows.html

Functional Preprocessing



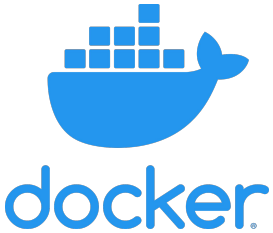


+ICA AROMA

No smoothing!

How does fMRIprep use all these softwares?

Containers:



Nipype, a python pipeline wrapper:

Containers

Containers provide an isolated software environment for a program and its dependencies. It's similar in some ways to the virtual machines that get set up when you use a HPC cluster but uses hardware differently.

As a user of containers, it's a two step process:

1. Write the container
2. Access the container environment

Installing fMRIprep on a server using singularity

1. Write the container

```
$ singularity build /my_images/fmriprep-<version>.simg docker://nipreps/fmriprep:<version>
```

I've had trouble sometimes with giving a version number to fMRIprep. You can also call this without a version number (i.e. build /my_images/fmriprep.simg docker://nipreps/fmriprep), but this is going to make it very hard to keep track of which version of fMRIprep you're using!

Running fMRIprep on a server

2. Access the container environment

The fMRIprep developers have provided a slurm script that makes it easy to get started on a slurm based cluster. I'm going to go through the sbatch script and show some potential places where errors can occur.

```
export STUDY=/path/to/some/folder
sbatch --array=1-$(( $( wc -l $STUDY/data/participants.tsv | cut -f1 -d' ' ) - 1 )) sbatch.slurm
```

I've titled my file sbatch_fmriprep to help distinguish it from the script on the fMRIprep site

```
#!/bin/bash
#SBATCH --nodes=1
#SBATCH --time=48:00:00
#SBATCH --job-name=fmriprep_44
#SBATCH --partition=netsi_standard
#SBATCH --mem-per-cpu=5Gb
#SBATCH --cpus-per-task=4

#SBATCH -o /scratch/sebruf/%x-%A-%a.out
#SBATCH -e /scratch/sebruf/%x-%A-%a.err
#SBATCH --mail-user=s.ruf@northeastern.edu
#SBATCH --mail-type=ALL

module load singularity/3.4.2

BIDS_DIR="$STUDY"
DERIVS_DIR="derivatives/fmriprep"
LOCAL_FREESURFER_DIR="$STUDY/derivatives/freesurfer-6.0.1"

# Prepare some writeable bind-mount points.
TEMPLATEFLOW_HOST_HOME=$HOME/.cache/templateflow
FMRIPREP_HOST_CACHE=$HOME/.cache/fmriprep
mkdir -p ${TEMPLATEFLOW_HOST_HOME}
mkdir -p ${FMRIPREP_HOST_CACHE}

# Prepare derivatives folder
mkdir -p ${BIDS_DIR}/${DERIVS_DIR}

# Make sure FS_LICENSE is defined in the container.
export SINGULARITYENV_FS_LICENSE=$HOME/.freesurfer.txt

# Designate a templateflow bind-mount point
export SINGULARITYENV_TEMPLATEFLOW_HOME="/templateflow"
```

Worth experimenting with a bit

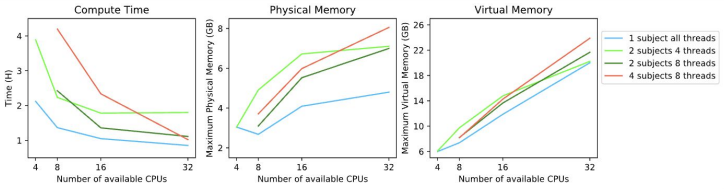
Use same version as when setting up fMRIprep

Make sure you have the license in the right place

How much CPU time and RAM should I allocate for a typical fMRIPrep run?

The recommended way to run fMRIPrep is to process one subject per container instance. A typical preprocessing run without surface processing with FreeSurfer can be completed in about 2 hours with 4 CPUs or in about 1 hour with 16 CPUs. More than 16 CPUs do not translate into faster processing times for a single subject. About 8GB of memory should be available for a single subject preprocessing run.

Below are some benchmark data that have been computed on a high performance cluster compute node with Intel E5-2683 v4 CPUs and 64 GB of physical memory:



Compute Time: time in hours to complete the preprocessing for all subjects. **Physical Memory**: the maximum of RAM usage used across all fMRIPrep processes as reported by the HCP job manager. **Virtual Memory**: the maximum of virtual memory used across all fMRIPrep processes as reported by the HCP job manager. **Threads**: the maximum number of threads per process as specified with `--omp-nthreads` in the fMRIPrep command.

The above figure illustrates that processing 2 subjects in 2 fMRIPrep instances with 8 CPUs each is approximately as fast as processing 2 subjects in one fMRIPrep instance with 16 CPUs. However, on a distributed compute cluster, the two 8 CPU instances may be allocated faster than the single 16 CPU instance, thus completing faster in practice. If more than one subject is processed in a single fMRIPrep instance, then limiting the number of threads per process to roughly the number of CPUs divided by the number of subjects is most efficient.

Setting up the container

In the container, fMRIprep is going to be looking for files in specific directories,like /data and /work. Here we're mainly defining those directories

Check the version

```
SINGULARITY_CMD="singularity run --home $HOME --cleanenv -B $BIDS_DIR:/data -B ${TEMPLATEFLOW_HOST_HOME}:${SINGULARITYENV_TEMPLATEFLOW_HOME} -B /scratch/sebruf/fmriprep_files:/work -B ${LOCAL_FREESURFER_DIR}:/fsdir $HOME/fmriprep-20.1.1.simg"
```

All the intermediate files are going to be stored here, make sure you have space for that

The fMRIprep command

```
# Parse the participants.tsv file and extract one subject ID from the line corresponding to this SLURM task.
subject=$( sed -n -E "$((${SLURM_ARRAY_TASK_ID} + 1))s/sub-(\S*)\>.*/\1/gp" ${BIDS_DIR}/participants.tsv )
#export subject=03
# Remove IsRunning files from FreeSurfer
find ${LOCAL_FREESURFER_DIR}/sub-$subject/ -name "*IsRunning*" -type f -delete

# Compose the command line
cmd="${SINGULARITY_CMD} /data /data/${DERIVS_DIR} participant --participant-label $subject -w /work/ -v --omp-nthreads 8 --nthreads 12 --mem_mb 160000 --output-spaces MNI152NLin2009cAsym:res-2 anat fsnative fsaverage5 --skull-strip-t1w force --use-aroma --fs-subjects-dir /fsdir"
```

Handles interrupted runs

Sets max computation per process

Sets options for fMRIprep

Clean up and exit

```
# Setup done, run the command
echo Running task ${SLURM_ARRAY_TASK_ID}
echo Commandline: $cmd
eval $cmd
exitcode=$?

# Output results to a table
echo "sub-$subject   ${SLURM_ARRAY_TASK_ID}    $exitcode" \
      >> ${SLURM_JOB_NAME}.${SLURM_ARRAY_JOB_ID}.tsv
echo Finished tasks ${SLURM_ARRAY_TASK_ID} with exit code $exitcode
exit $exitcode
```

Preparing Data

Brain Imaging Data Structure (BIDS) format. fMRIprep does the rest automatically

For lesion masks:

Cost function masking during spatial normalization

When processing images from patients with focal brain lesions (e.g., stroke, tumor resection), it is possible to provide a lesion mask to be used during spatial normalization to standard space [Brett2001]. ANTs will use this mask to minimize warping of healthy tissue into damaged areas (or vice-versa). Lesion masks should be binary NIfTI images (damaged areas = 1, everywhere else = 0) in the same space and resolution as the T1 image, and follow the naming convention specified in BIDS Extension Proposal 3: Common Derivatives (e.g., `sub-001_T1w_label-lesion_roi.nii.gz`). This file should be placed in the `sub-*/anat` directory of the BIDS dataset to be run through *fMRIPrep*. Because lesion masks are not currently part of the BIDS specification, it is also necessary to include a `.bidsignore` file in the root of your dataset directory. This will prevent bids-validator from complaining that your dataset is not valid BIDS, which prevents *fMRIPrep* from running. Your `.bidsignore` file should include the following line:

```
*lesion_roi.nii.gz
```

fMRIprep Outputs: Files

Structural

```
sub-<subject_label>/
  anat/
    sub-<subject_label>[_space-<space_label>]_desc-preproc_T1w.nii.gz
    sub-<subject_label>[_space-<space_label>]_desc-brain_mask.nii.gz
    sub-<subject_label>[_space-<space_label>]_dseg.nii.gz
    sub-<subject_label>[_space-<space_label>]_label-CSF_probseg.nii.gz
    sub-<subject_label>[_space-<space_label>]_label-GM_probseg.nii.gz
    sub-<subject_label>[_space-<space_label>]_label-WM_probseg.nii.gz
```

Functional

```
sub-<subject_label>/
  func/
    sub-<subject_label>_[specifiers]_space-<space_label>_boldref.nii.gz
    sub-<subject_label>_[specifiers]_space-<space_label>_desc-brain_mask.nii.gz
    sub-<subject_label>_[specifiers]_space-<space_label>_desc-preproc_bold.nii.gz
```

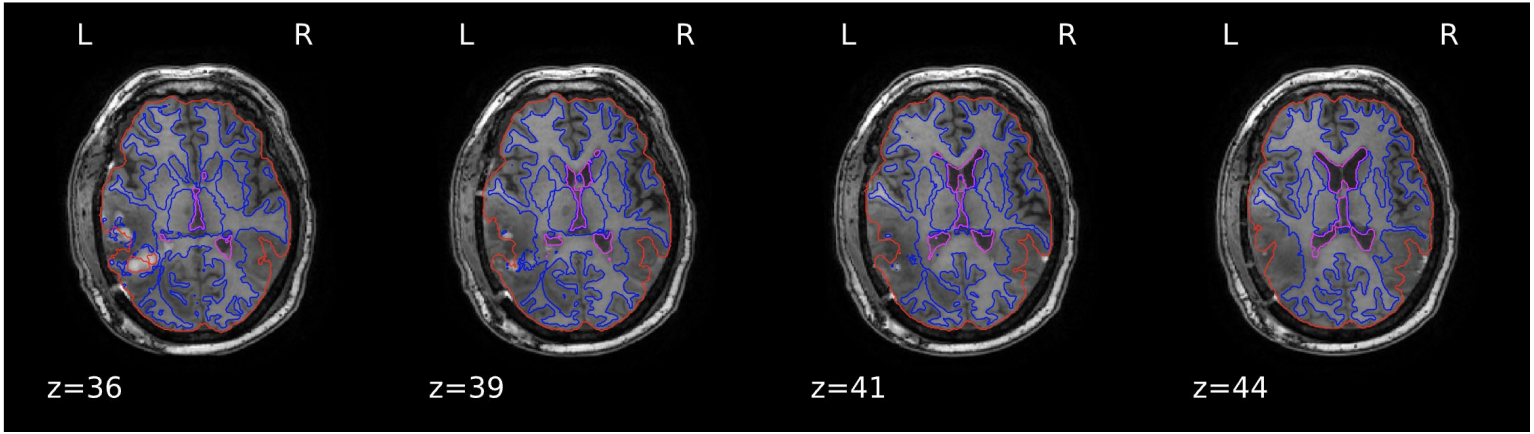
Confounds

```
sub-<subject_label>/
  func/
    sub-<subject_label>_[specifiers]_desc-confounds_timeseries.tsv
    sub-<subject_label>_[specifiers]_desc-confounds_timeseries.json
```

QA plots

Brain mask and brain tissue segmentation of the T1w

This panel shows the template T1-weighted image (if several T1w images were found), with contours delineating the detected brain mask and brain tissue segmentations.



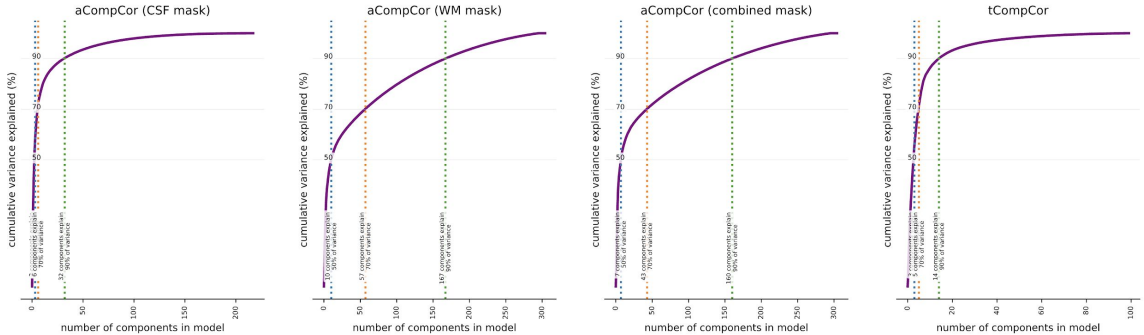
I just pulled some examples, there's a lot more in the QA plots. You can find more info here:

https://fmriprep.org/en/latest/outputs.html

QA Confounds

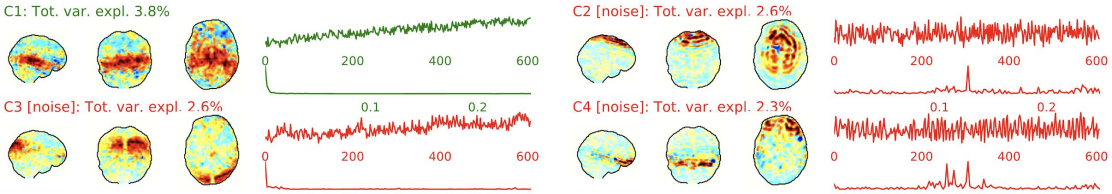
Variance explained by t/aCompCor components

The cumulative variance explained by the first k components of the *t/aCompCor* decomposition, plotted for all values of *k*. The number of components that must be included in the model in order to explain some fraction of variance in the decomposition mask can be used as a feature selection criterion for confound regression.



ICA Components classified by AROMA

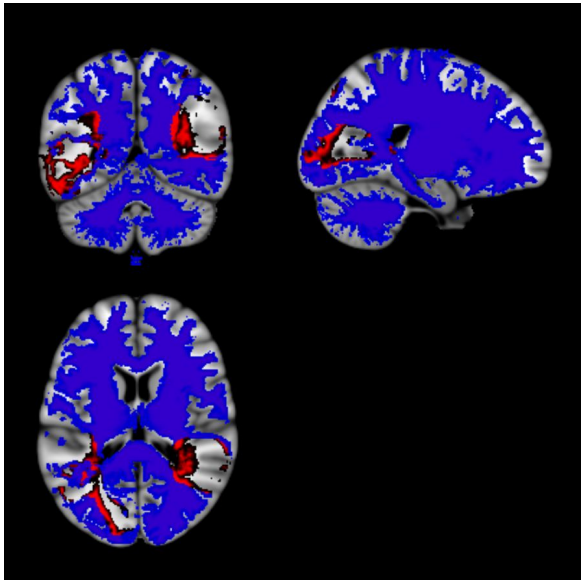
Maps created with maximum intensity projection (glass brain) with a black brain outline. Right hand side of each map: time series (top in seconds), frequency spectrum (bottom in Hertz). Components classified as signal are plotted in green; noise components in red.



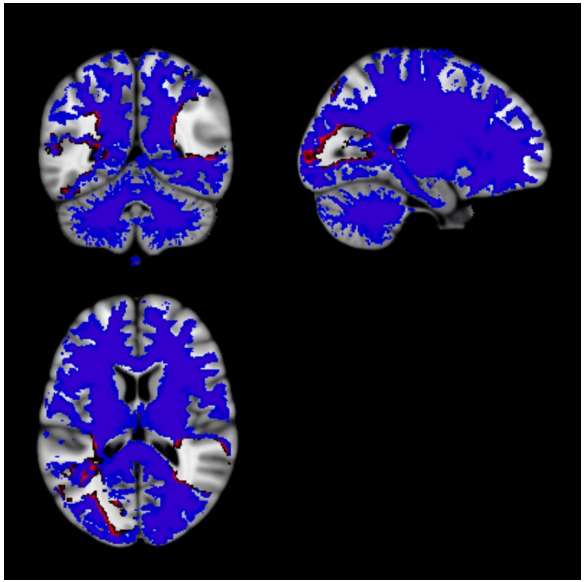
Segmentation

fMRIprep only deals with lesion information in the normalization step! If you want to be careful about how you’re dealing with the lesion mask, this is going to cause problems.

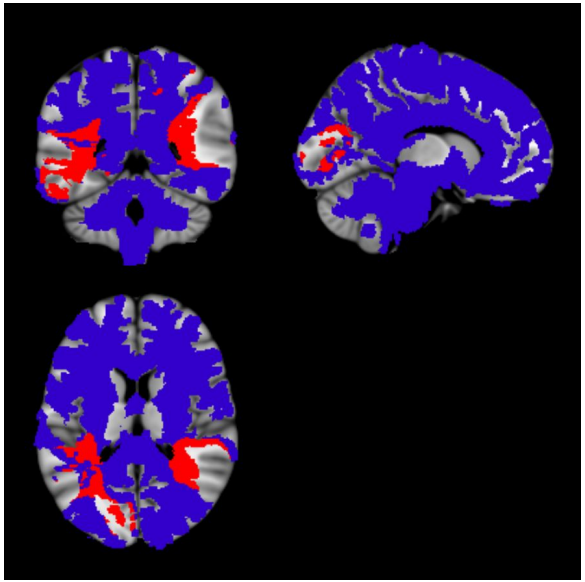
White Matter with Red Lesion Overlap



SPM 12 Unified



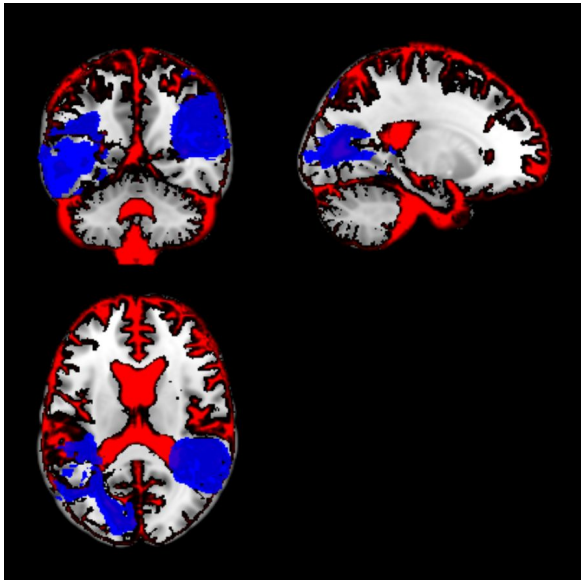
SPM12 + Modified TPM



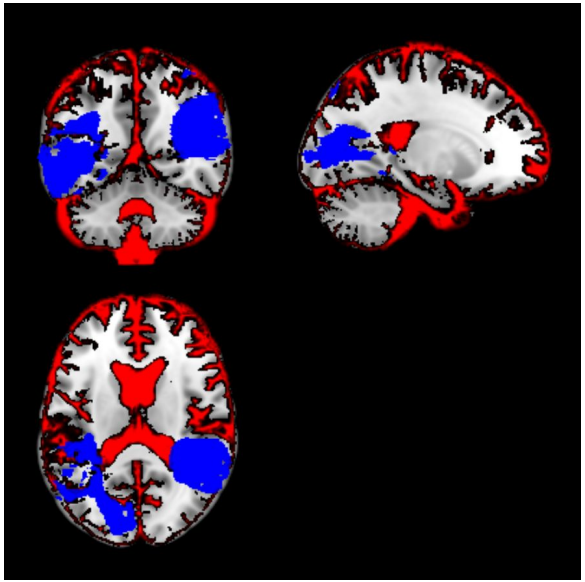
fMRIprep (binarized)

fMRIprep white matter masks tend to have high overlap with lesion tissue

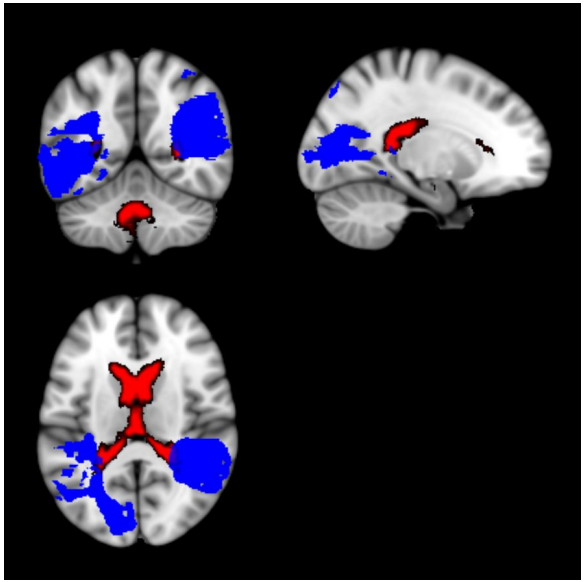
CSF



SPM 12 Unified



SPM12 + Modified TPM



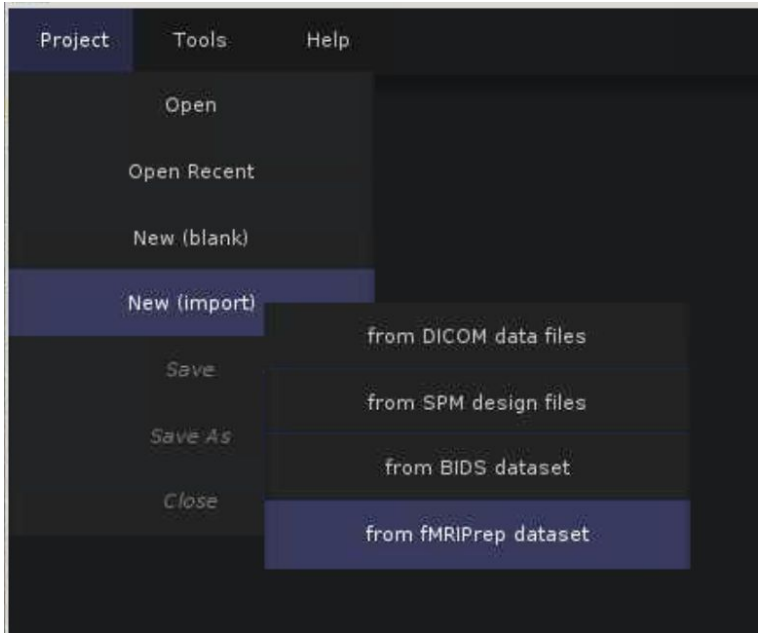
fMRIprep

fMRIprep has conservative CSF estimates. In most cases this works in our favor for reducing overlap with lesion

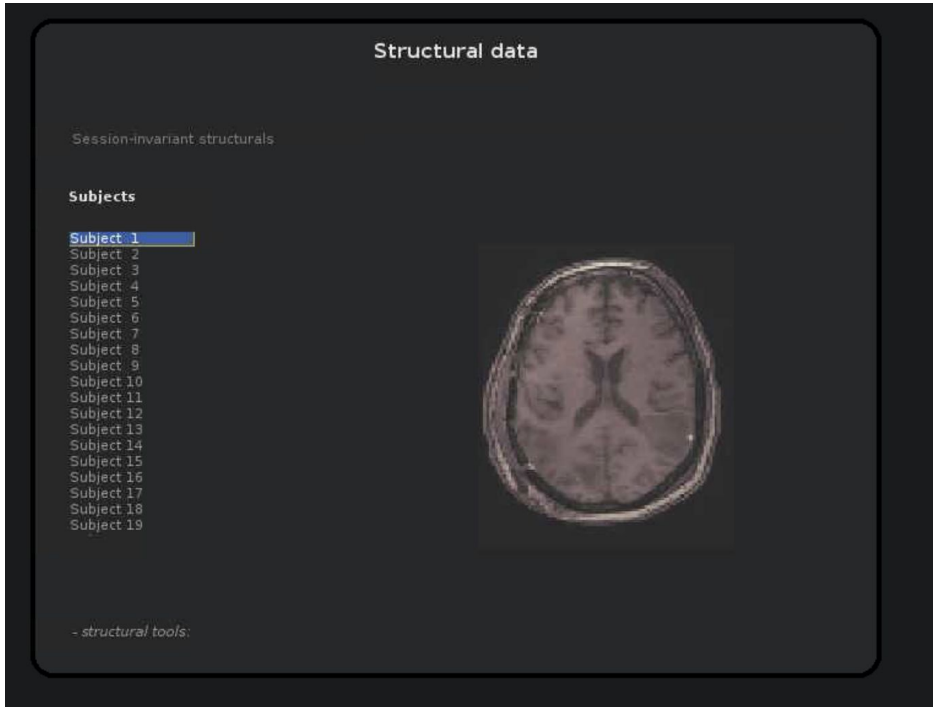
Loading into CONN

I’ve included both how to do it and things that are going to be different from a standard CONN run.

Loading into CONN

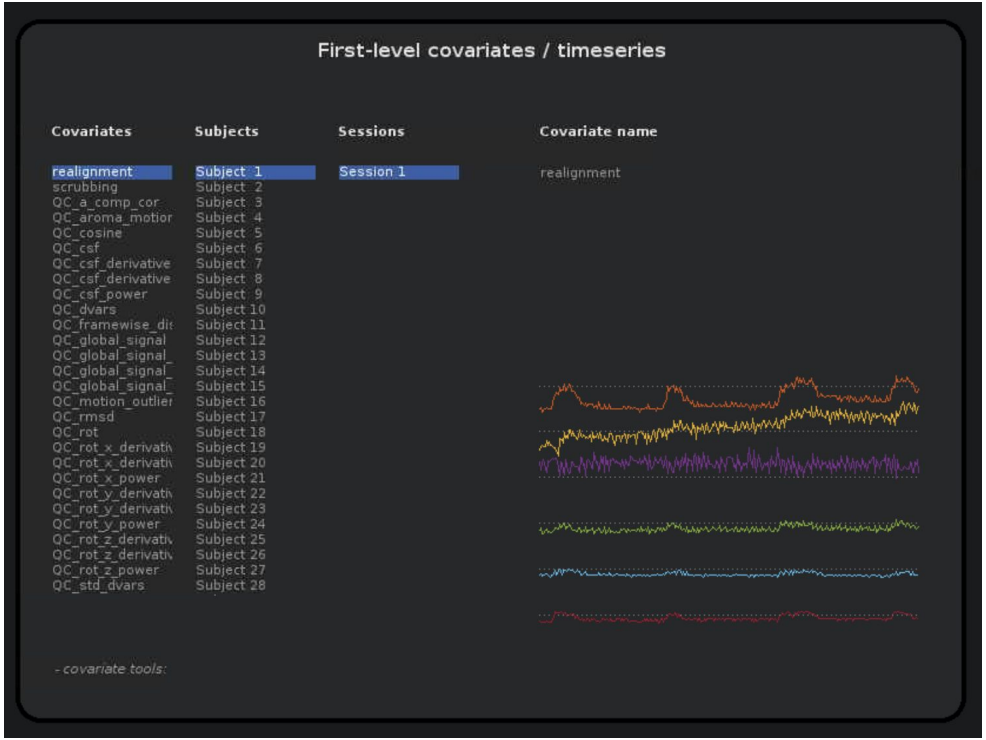


Different: fMRIprep does not save skull stripped T1w images



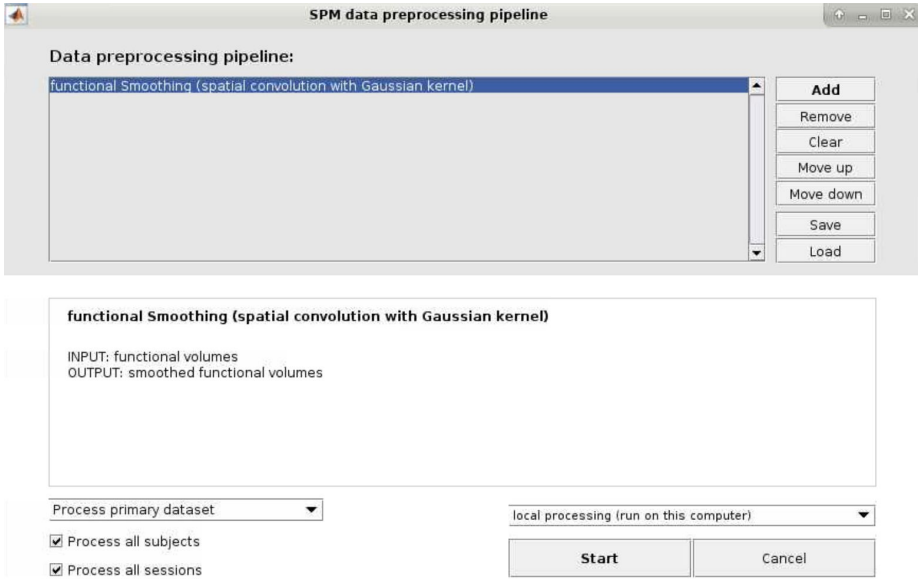
Different: Covariates

Conn loads in the covariates automatically, they are very similar to what gets calculated in CONN



Different: Smoothing

If you want smoothing...

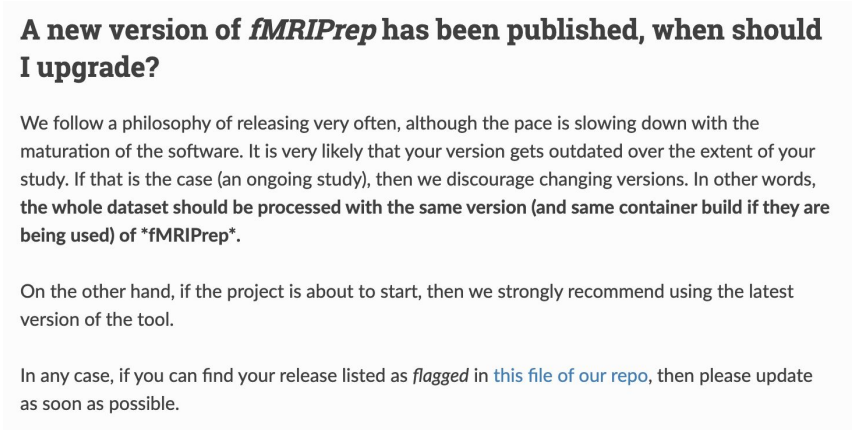


Otherwise you just have to click done in the Setup Tab

fMRIprep in context

It’s a powerful tool once it’s set up.

It only uses lesion information in the normalization step though, so it’s going to depend on your use case



A new version of *fMRIPrep* has been published, when should I upgrade?

We follow a philosophy of releasing very often, although the pace is slowing down with the maturation of the software. It is very likely that your version gets outdated over the extent of your study. If that is the case (an ongoing study), then we discourage changing versions. In other words, **the whole dataset should be processed with the same version (and same container build if they are being used) of *fMRIPrep*.**

On the other hand, if the project is about to start, then we strongly recommend using the latest version of the tool.

In any case, if you can find your release listed as *flagged* in this file of our repo, then please update as soon as possible.