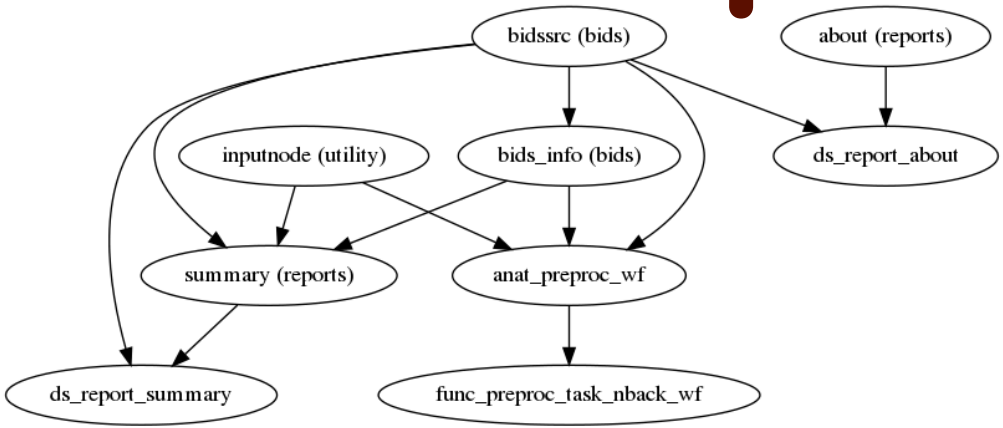
WTfMRIprep?

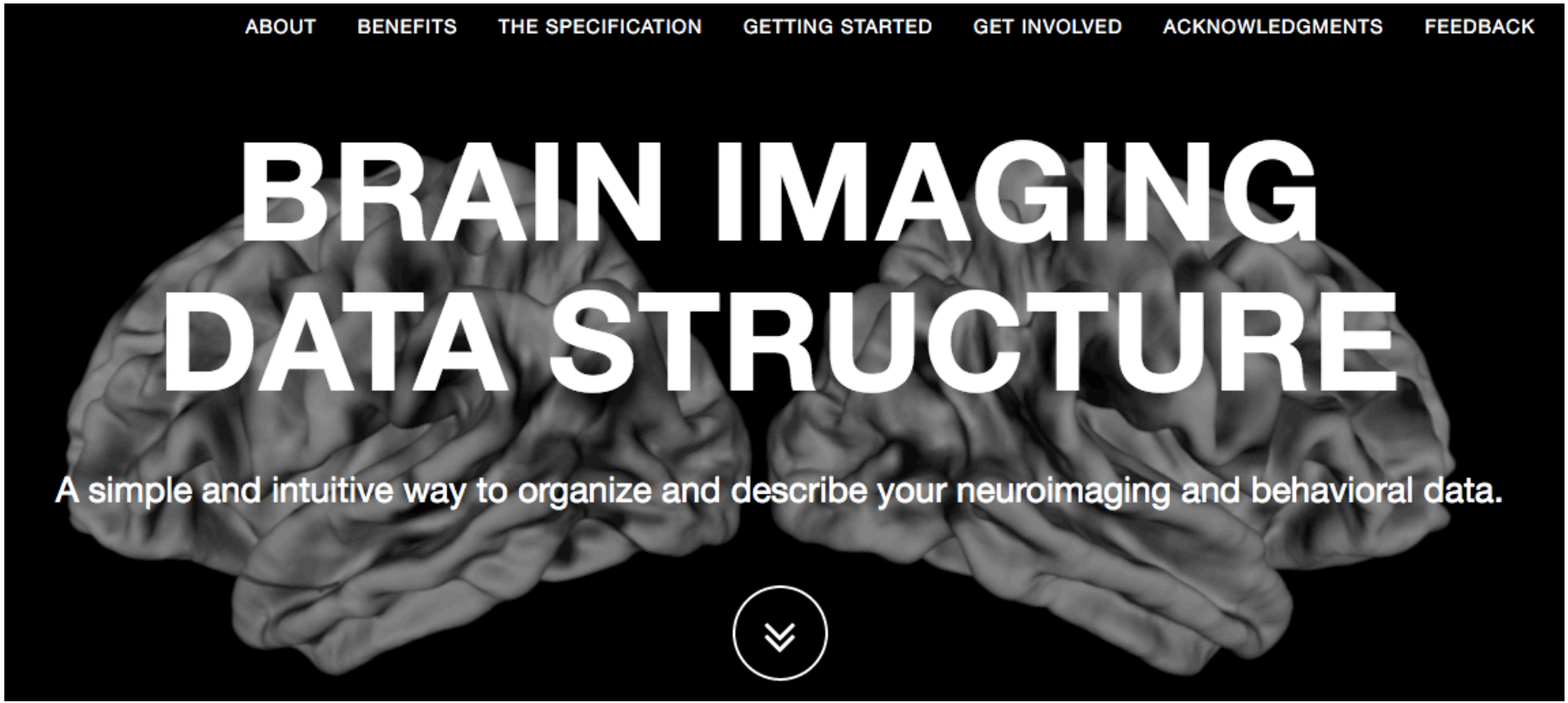


An attempt to demystify what the freak fMRIprep is, does, and how (with a long discussion of BIDS).

Outline

- Brain Imaging Data Structure (BIDS)
  - What and why
  - Converting to BIDS
- fMRIprep
  - What
  - Why
  - How
  - Outputs

BIDS: https://bids.neuroimaging.io/



BIDS: https://bids.neuroimaging.io/

About BIDS

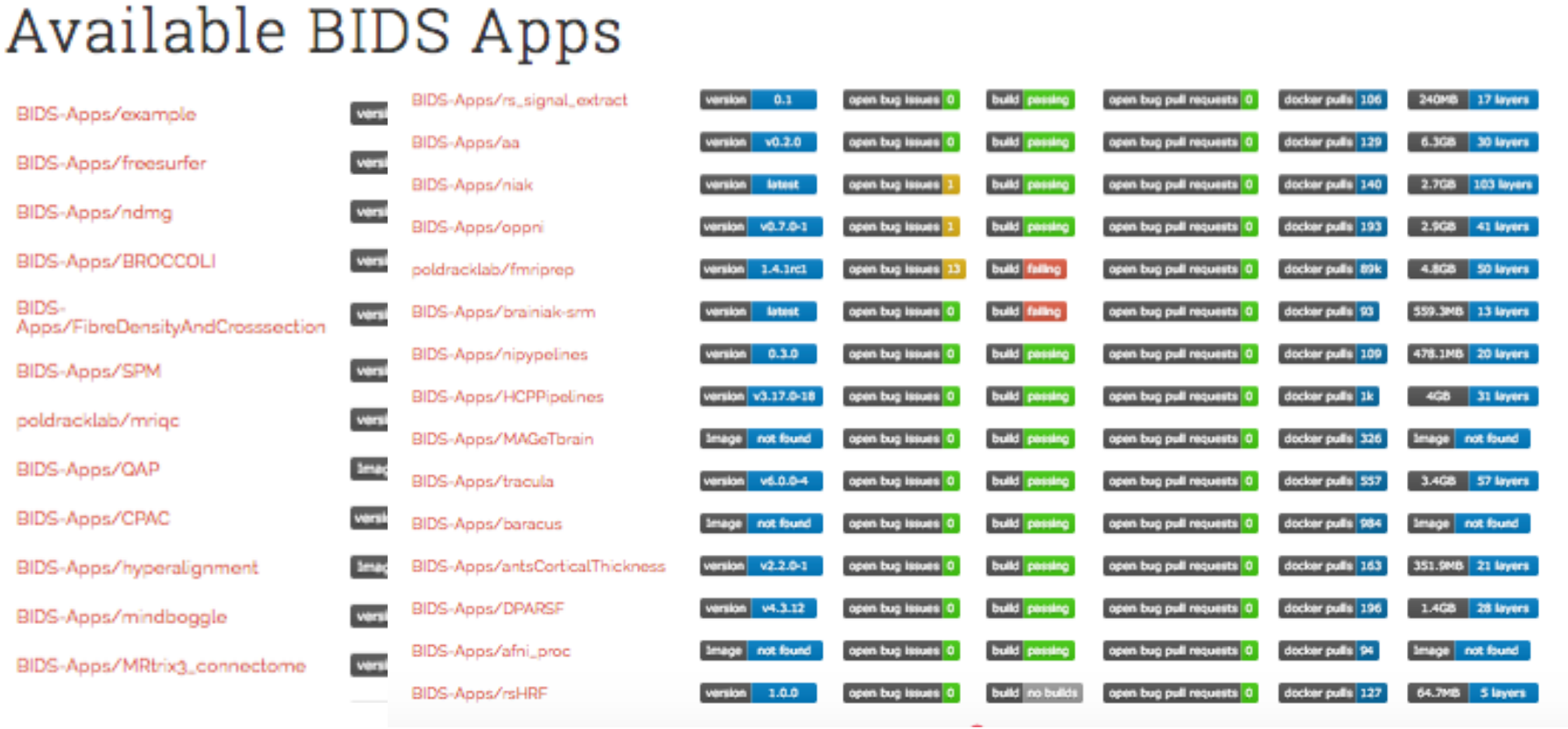
“Neuroimaging experiments result in complicated data that can be arranged in many different ways. So far there is no consensus how to organize and share data obtained in neuroimaging experiments. Even two researchers working in the same lab can opt to arrange their data in a different way. Lack of consensus (or ***a standard***) leads to misunderstandings and time wasted on rearranging data or rewriting scripts expecting certain structure. Here we describe a simple and easy to adopt way of organizing neuroimaging and behavioral data.”

BIDS: WHY (TF are you boring us with data organization)?

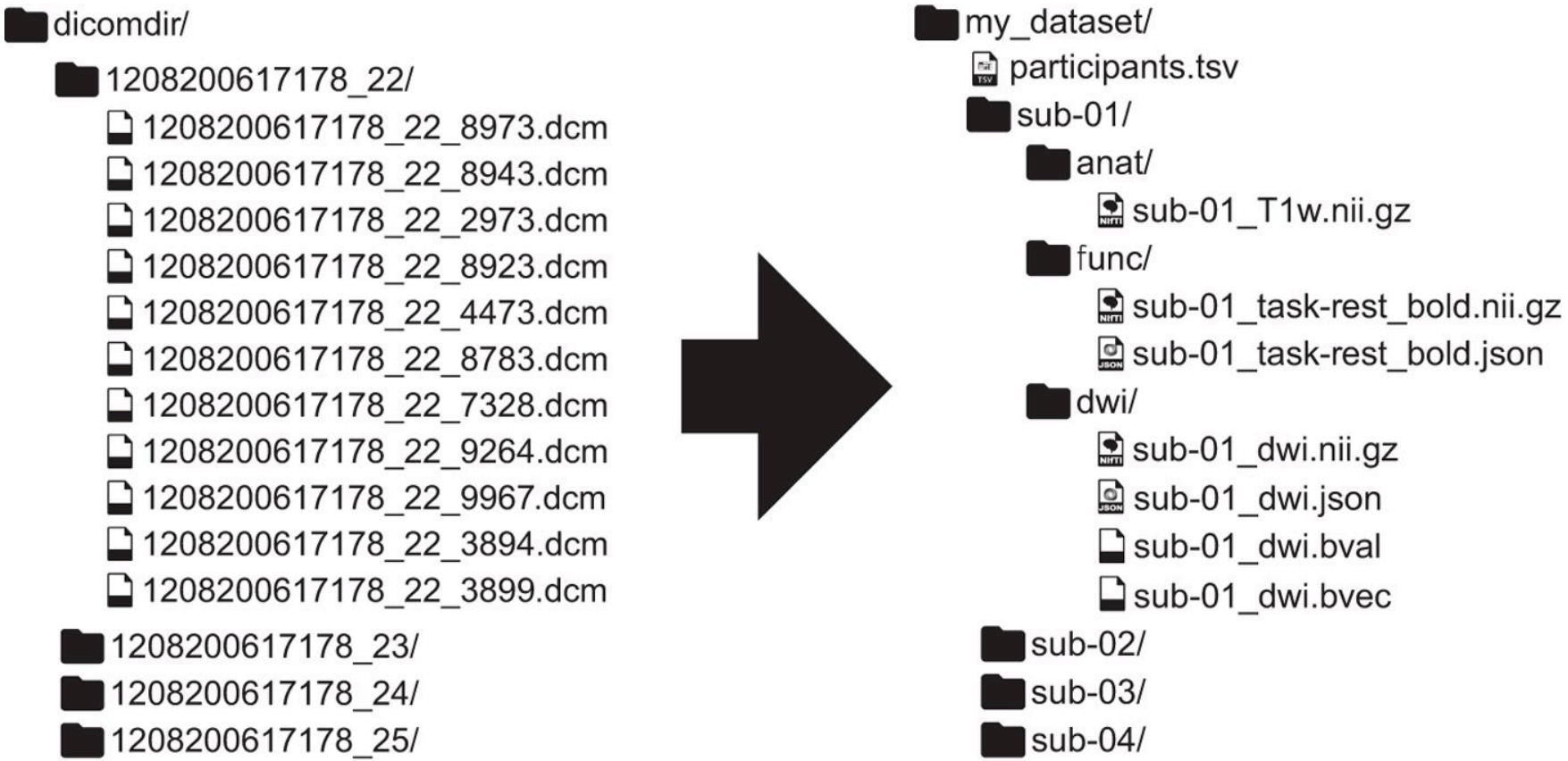
Standardization of how brain imaging data is (or should be) organized!

- Reproducibility & replicability
- Better documentation
- Easy data sharing
  - OpenNeuro.org (& future of fMRI databases at NIH!)
  - Open science best practices
- Machine readable = easier automation!
  - So many plug & play BIDS compatible apps (like fmriprep;)

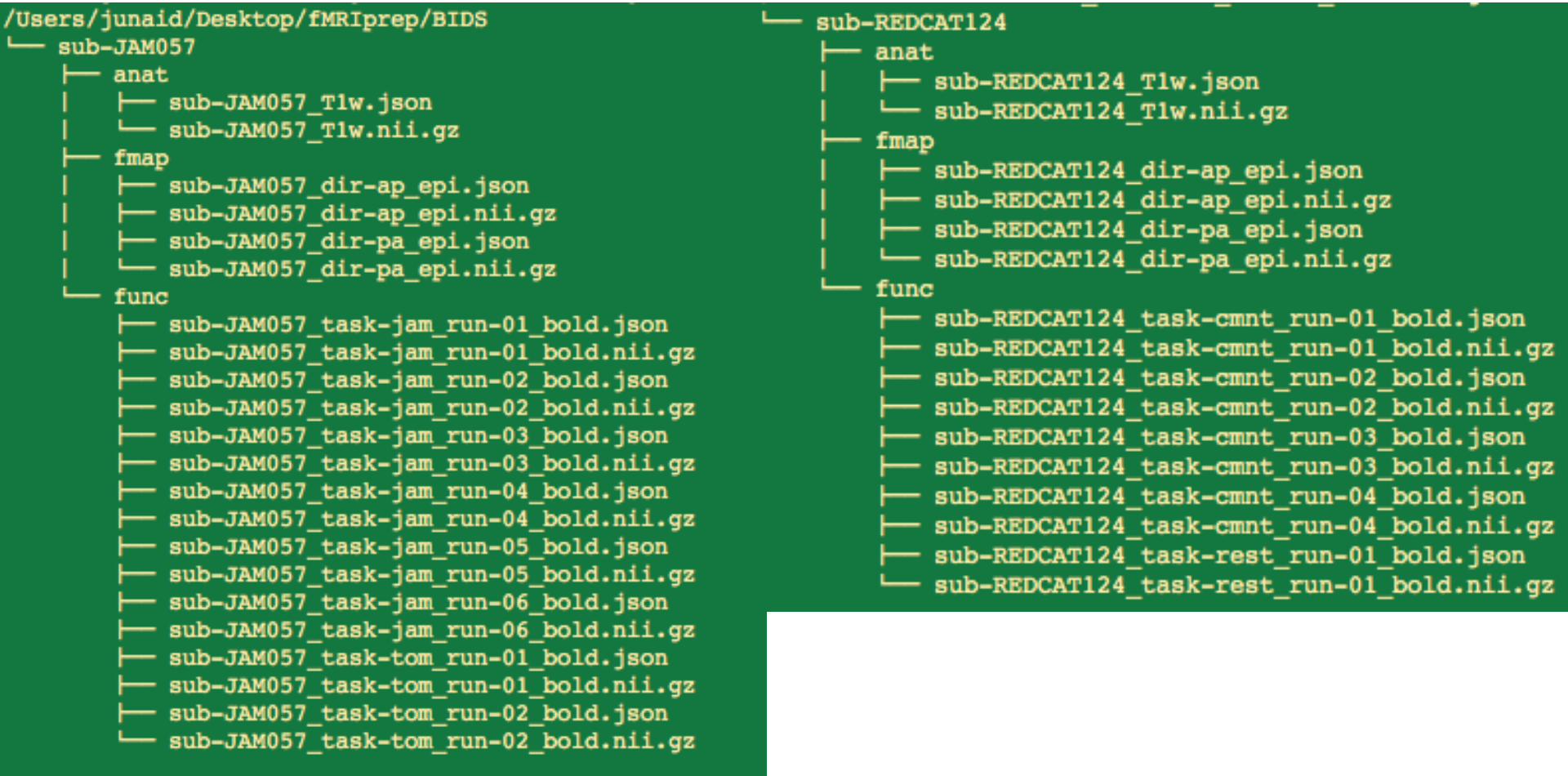
BIDS Apps: http://bids-apps.neuroimaging.io/



BIDS: What does it look like?



BIDS: What does *our's* look like?



BIDS: What does *our's* look like?

So, this is not exactly a fully BIDS compatible data-set. BIDS requires a couple more things:

- A dataset_description.json file at the super-level directory
- TaskName section in .json files for the jam .json files: “You have to define 'TaskName' for this file. It can be included one of the following locations: /sub-JAM057/func/sub-JAM057_task-jam_bold.json, /sub-JAM057/sub-JAM057_task-jam_bold.json, /task-jam_bold.json, /sub-JAM057/func/sub-JAM057_task-jam_run-01_bold.json” (from BIDS validator)

Other things needed but not absolutely required:

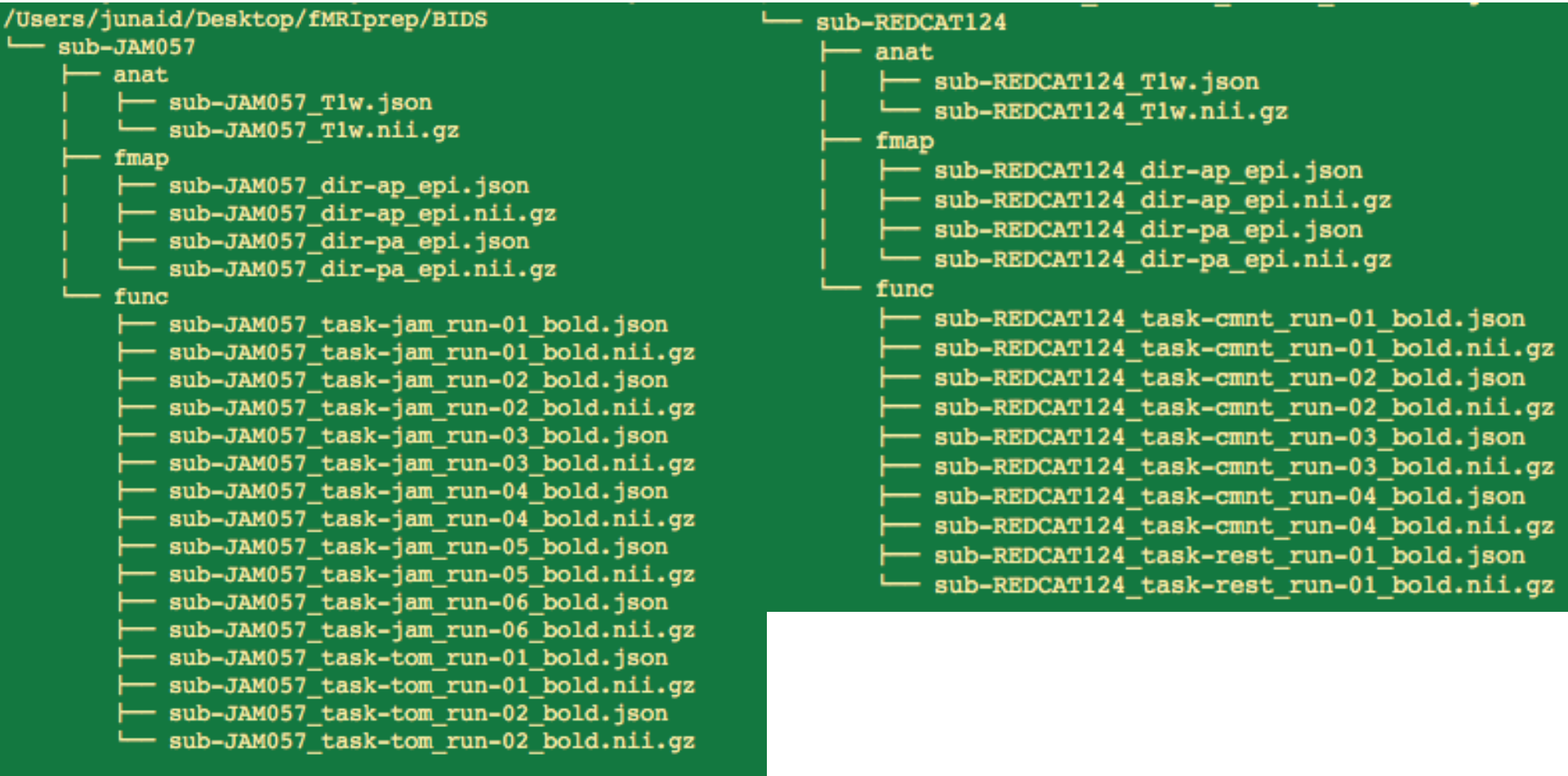
- .tsv files with the timing parameters for non-rest scans
- And a README file at the super-level directory

BUT, this not-fully BIDS compatible data-set still works with fMRIprep!

Minimal BIDS requirements to work with fMRIprep:

- Subject level directories labeled as such: sub-<subjectid>
  - <subjectid> cannot have dashes, spaces, underscores, or special characters
- Within this directory, there are directories for each MR modality:
  - /anat - which contains all structural/anatomical scans
  - /func - all functional MRI scans including task and rest
  - /fmap - fieldmap scans. Can handle double-echo and opposite phase encoding scans. fMRIprep even has a fieldmapless distortion correction option!
  - /dwi - diffusion weighted scans if you have it i.e. DTI
  - There are other things they are expanding this to also--MEG, EEG etc
- !! Each MRI file needs accompanying .json file (with same name) that has all info about the scan !!
- All MRI files need to be in gzipped nifti format, that is **.nii.gz**, and 4D when applicable (i.e. functional runs)
- Don’t freak out yet; most of this is handled by dicom conversion software/scripts!

BIDS: What does *our's* look like?



BIDS naming conventions (focusing on our data):

All MRI file names start first with the sub-<subjectid>, followed by an underscore. E.g. **sub-JAM057_** (the underscore & dash conventions make it machine readable).

- anat: for our T1 MPRAGE structural, the scan files are simply **T1w.nii.gz**.
  - E.g. **sub-JAM057_T1w.nii.gz**
- func: functional files are **task-<taskname or rest>_run-<runnumber>_bold.nii.gz**
  - E.g. **sub-JAM057_task-jam_run-01_bold.nii.gz sub-REDCAT124_task-cmnt_run-01_bold.nii.gz & sub-REDCAT124_task-rest_run-01_bold.nii.gz**
  - No dashes, underscores, spaces, or special characters in the task name
  - Run number is always two digits, if included (not necessary).
  - Other specifications for longitudinal data too but not covered here.
- fmap: for the opposite phase-encoding direction fieldmaps (that we use), just follow the examples.

Not sure the minimal information required in the accompanying .json file for each, but conversion software provides all the scanning parameters and more!

BIDS Conversion using my dcm2niix wrapper

dcm2niix is dicom to nifti converter that provides the accompanying .json file you need for BIDS compatibility: https://github.com/rordenlab/dcm2niix

To match the BIDS naming convention & specifications, I created a pair of scripts that ‘wrap’ around the basic dcm2niix function.

BidsConvert.sh is the main script that takes the parameters file as input, and does the conversion.

BidsConvertParameters.sh is the parameters file that you edit to fit the needs of your study.

Usage: **bash /path/to/BidsConvert.sh /path/to/BidsConvertParameters.sh**

BIDS Conversion dcm2niix wrapper: Parameters file

```
#!/bin/bash

# This is the bash parameters file to be used in conjunction with BidsConvert.sh
# Modify everything in quotes for each element (1-7) below for your study.

# 1) Subjects
# Define the subject IDs for the data you want to convert. These IDs should
# match the IDs of the dicom folders. You can include numerous subjects or just
# one. It is OK if the subject IDs have dashs, underscores, or special
# characters because the BidsConvert script will remove them from the name.
export SubID=("RED_CAT_124" "RED_CAT_125" "RED_CMNT_125")

# 2) Dicom Directory
# Define the super directory where all the dicom folders reside.
export DcmDir="/data/bswift-1/jmerch/CAT/dicoms"

# 3) BIDS Study Directory
# Define the super directory where you want your BIDS organized data to reside.
export OutDir="/data/bswift-1/jmerch/CAT/in"

# 4) Path To dcm2niix
# Define path to the dcm2niix script. If dcm2niix is added to the bash/tcsh
# path/environment, you can simply use:
# Vert="dcm2niix"
export Vert="/data/bswift-1/jmerch/JAM/dcm2niix"

# 5) Functional Scans
# Define the names of the functional dicom folders to convert.
export FuncDcms=("cmnt1" "cmnt2" "cmnt3" "cmnt4" "fix")

# Using the same order as above, define what you want them to be named
# REMEBER: Bids format does not like dashs, underscores, or any special
# characters for the name of the functional files. I incorporated this because
# the functional dicom folder names were not very descriptive.
export FuncName=("cmnt_run-01" "cmnt_run-02" "cmnt_run-03" "cmnt_run-04" "rest_run-01")

# 6) Structural Scans
# Define the names of the structural dicom folders to convert.
# Right now, this set up only allows for T1/MPRAGE structurals.
export StrctDcms=("t1_mpr")

# 7) Fieldmap Scans
# Define the names of the fieldmap dicom folders to convert.
# Right now, this only allows for opposite phase encoding direction fmaps.
export FmapDcms=("AP" "PA")
```

BIDS Conversion dcm2niix wrapper: Parameters file

What you need to edit:

1. Subject IDs as they are in the dicom directory. You can include dashes/underscores because the main script will remove them.
2. Path to dicom directory.
3. Path to directory where you want the converted files written.
4. Path to dcm2niix file (it's a stand alone application but requires full path).
5. The names of the functional folders as they are in the dicom directory.
6. The names you want the functional files to have when converted, following the same order as the items in 5 above. Use BIDS specifications here!!
7. The name of the structural folder as it is in the dicom directory.
8. The names of the fieldmap folders as they are in the dicom directory.

There is one hacky thing in the main script to make fieldmaps work that needs work..

```
# 1) Subjects
# Define the subject IDs for the data you want to convert. These IDs should
# match the IDs of the dicom folders. You can include numerous subjects or just
# one. It is OK if the subject IDs have dashs, underscores, or special
# characters because the BidsConvert script will remove them from the name.
export SubID=("RED_CAT_124" "RED_CAT_125" "RED_CMNT_125")

# 2) Dicom Directory
# Define the super directory where all the dicom folders reside.
export DcmDir="/data/bswift-1/jmerch/CAT/dicoms"

# 3) BIDS Study Directory
# Define the super directory where you want your BIDS organized data to reside.
export OutDir="/data/bswift-1/jmerch/CAT/in"

# 4) Path To dcm2niix
# Define path to the dcm2niix script. If dcm2niix is added to the bash/tcsh
# path/environment, you can simply use:
# Vert="dcm2niix"
export Vert="/data/bswift-1/jmerch/JAM/dcm2niix"

# 5) Functional Scans
# Define the names of the functional dicom folders to convert.
export FuncDcms=("cmnt1" "cmnt2" "cmnt3" "cmnt4" "fix")

# Using the same order as above, define what you want them to be named
# REMEBER: Bids format does not like dashs, underscores, or any special
# characters for the name of the functional files. I incorporated this because
# the functional dicom folder names were not very descriptive.
export FuncName=("cmnt_run-01" "cmnt_run-02" "cmnt_run-03" "cmnt_run-04" "rest_run-01")

# 6) Structural Scans
# Define the names of the structural dicom folders to convert.
# Right now, this set up only allows for T1/MPRAGE structurals.
export StrctDcms=("t1_mpr")

# 7) Fieldmap Scans
# Define the names of the fieldmap dicom folders to convert.
# Right now, this only allows for opposite phase encoding direction fmaps.
export FmapDcms=("AP" "PA")
```

Final things to check out regarding BIDS

Learn from example by looking at these bids-examples: https://github.com/bids-standard/bids-examples/

Or check out the datasets at https://openneuro.org/

You can validate if your data structure is BIDS compatible: https://bids-standard.github.io/bids-validator/

But, as I said before, if you follow my examples, it will not pass full bids compatibility!

BIDS Validator

BIDS

Summary	Available Tasks	Available Modalities
• 38 Files, 2.85GB • 2 - Subjects • 1 - Session		• T1w • bold • fieldmap

Your dataset is not a valid BIDS dataset.

view 2 errors in 13 files

view 3 warnings in 38 files

Download error log for BIDS

Click to view details on BIDS specification

If you have any questions please post on Neurostars

The source code for the validator can be found here

Challenge 1 – BIDS conversion

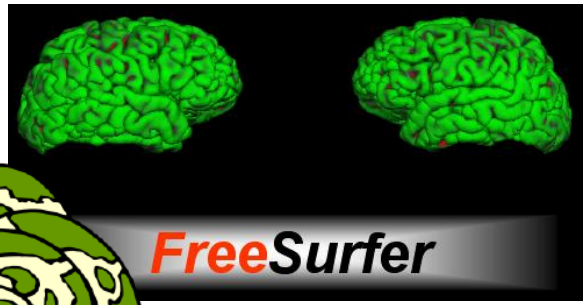
A) Download the zipped dicoms folder (and unzip) from this link (~2 GB): https://drive.google.com/file/d/15TKjVUXSxnCcYjUvB6CLmM_Mm20Xww-E/view?usp=sharing Then, download the dcm2niix application, the BidsConvert.sh & BidsConvertParameters.sh scripts from this repo. Now hack the BidsConvertParameters.sh file to match the paths of your machine, and run the script on the dicoms folder you downloaded. Document an errors that arise, and try to fix them.

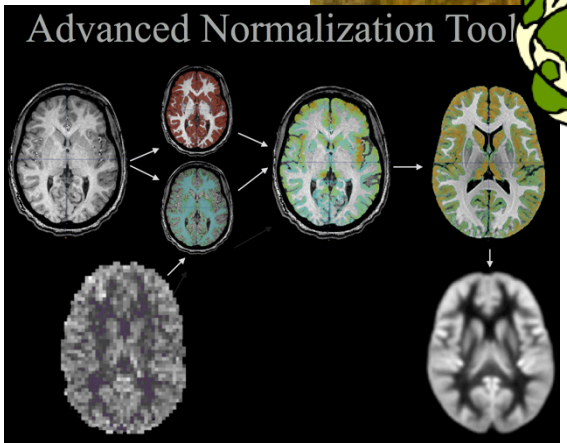
B) Use the BIDS validator to see if this converted dataset is BIDS compatible. Are there any errors or warnings? If so, what do they mean. Is this dataset ready for fmriprep? https://bids-standard.github.io/bids-validator/

C) BONUS: If you have a dicoms folder of your own data, modify the parameters file and try using the BIDS conversion scripts on your own data.

Time for fMRIprep: fmriprep.readthedocs.io/en/stable/

So, WTfMRIprep?!
fMRIprep is an advance fMRI preprocessing pipeline taking the best pieces from various different software packages, & pipes it together using NiPype python environment.





https://fmriprep.readthedocs.io/en/stable/

“fMRIprep is a functional magnetic resonance imaging (fMRI) data preprocessing pipeline that is designed to provide an easily accessible, state-of-the-art interface that is robust to variations in scan acquisition protocols and that requires minimal user input, while providing easily interpretable and comprehensive error and output reporting. It performs basic processing steps (coregistration, normalization, unwarping, noise component extraction, segmentation, skullstripping etc.) providing outputs that can be easily submitted to a variety of group level analyses, including task-based or resting-state fMRI, graph theory measures, surface or volume-based statistics, etc.”

https://fmriprep.readthedocs.io/en/stable/

This tool allows you to easily do the following:

- Take fMRI data from raw to fully preprocessed form.
- Implement tools from different software packages.
- Achieve optimal data processing quality by using the best tools available.
- Generate preprocessing quality reports, with which the user can easily identify outliers.
- Receive verbose output concerning the stage of preprocessing for each subject, including meaningful errors.
- Automate and parallelize processing steps, which provides a significant speed-up from typical linear, manual processing.

fMRIprep pipeline

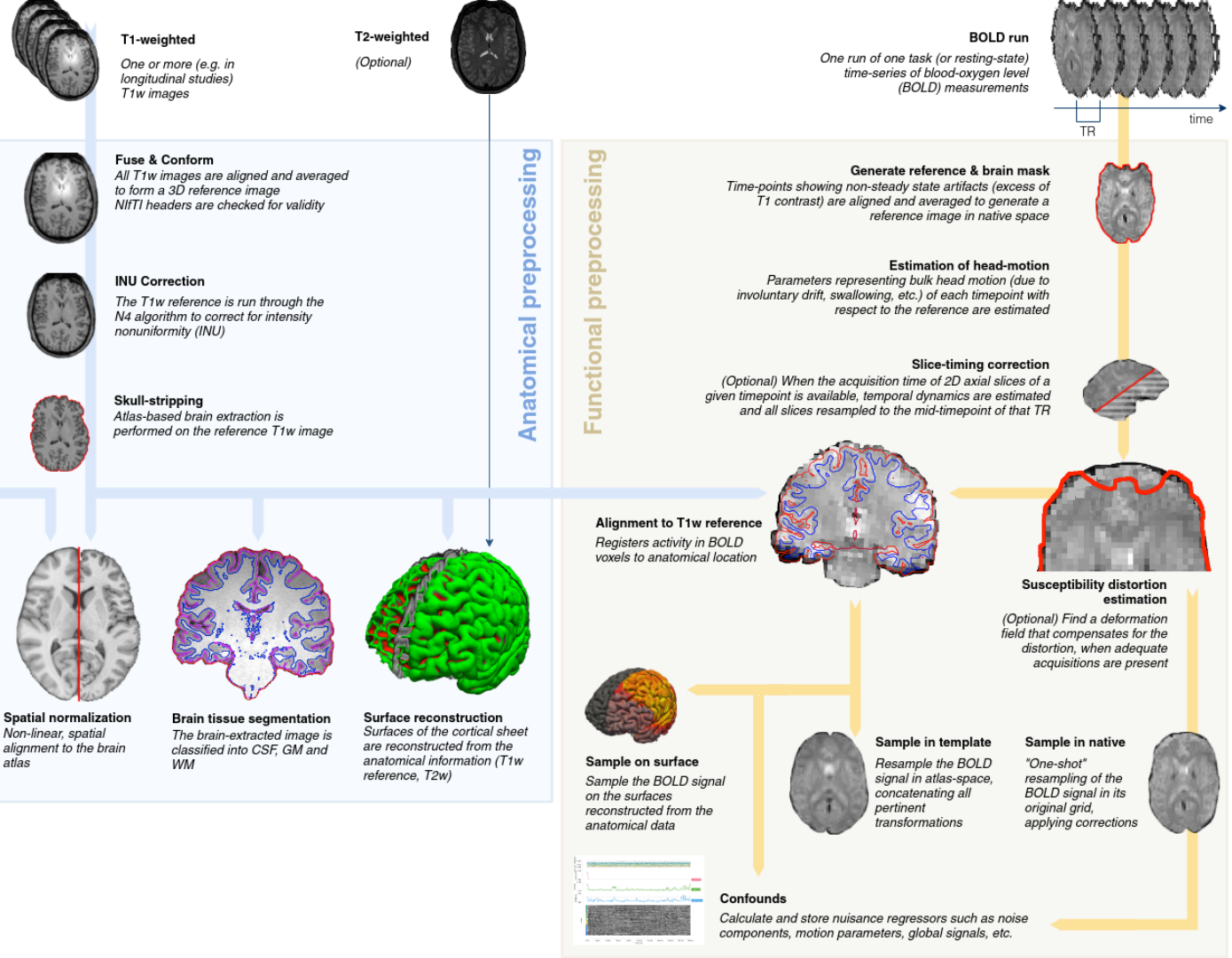


Table 1 | State-of-the-art neuroimaging offers a catalog of readily available software tools

Preprocessing task	Included with fMRIPrep	Alternatives (not included in fMRIPrep)
Anatomical T1-weighted brain extraction	antsBrainExtraction.sh (ANTs)	bet (FSL), 3dSkullstrip (AFNI), MRTOOL (SPM plug-in)
Anatomical surface reconstruction	recon-all (FreeSurfer)	CIVET, BrainSuite, Computational Anatomy (SPM plug-in)
Head-motion estimation (and correction)	MCFLIRT (FSL)	3dvolreg (AFNI), spm_realign (SPM), cross_realign_4dfp (4dfp), antsBrainRegistration (ANTs)
Susceptibility-derived distortion estimation (and unwarping)	3dqwarp (AFNI)	FUGUE and topup (FSL); FieldMap and HySCO (SPM plug-ins)
Slice-timing correction	3dTshift (AFNI)	slicetimer (FSL), spm_slice_timing (SPM), interp_4dfp (4dfp)
Intrasubject registration	bbregister (FreeSurfer), FLIRT (FSL)	3dvolreg (AFNI), antsRegistration (ANTs), Coregister (SPM GUI)
Spatial normalization (intersubject co-registration)	antsRegistration (ANTs)	@auto_tlrc (AFNI), FNIRT (FSL), Normalize (SPM GUI)
Surface sampling	mri_vol2surf (FreeSurfer)	SUMA (AFNI), MNE, Nilearn
Subspace projection denoising (e.g., independent or principal component analysis)	MELODIC (FSL), ICA-AROMA	Nilearn, LMGS (SPM plug-in)
Confounds	In-house implementation	fsl_motion_outliers (FSL), TAPAS PhysIO (SPM plug-in)
Detection of non-steady states	In-house implementation	Ad hoc implementations, manual setting

fMRIPrep integrates best-in-breed tools for each of the preprocessing tasks that its workflow covers, except for steps implemented as part of the development of fMRIPrep (in-house implementations). Tasks listed in the first column are described in detail in Supplementary Note 1.

fMRIprep Installation: https://fmriprep.readthedocs.io/en/stable/installation.html

4 ways to install/use fMRIprep:

1. Openneuro.org - Free and easy to use, but requires full dataset that you might not want to share yet
2. Docker container - Easiest in my experience, but is often not allowed on servers because of security issues
    a. http://reproducibility.stanford.edu/fmriprep-tutorial-running-the-docker-image/
3. **Singularity container - Similar to Docker, but allowed on servers. A little harder to use**
4. Manually prepared environment - Not recommended

Quick note on Singularity and containerized environments

- A containerized environment is like a virtual machine
- You can shell into it like a server
- You can gather everything required for fMRIprep in one packaged environment (single image file) that you can use on any computer/server
- You can map data/folders between your local environment and the containerized environment

https://www.sylabs.io/guides/2.6/user-guide/

fMRIprep usage:
https://fmriprep.readthedocs.io/en/stable/usage.html

```
usage: fmriprep [-h] [--version] [--skip_bids_validation]
                [--participant_label PARTICIPANT_LABEL [PARTICIPANT_LABEL ...]]
                [-t TASK_ID] [--echo-idx ECHO_IDX] [--nthreads NTHREADS]
                [--omp-nthreads OMP_NTHREADS] [--mem_mb MEM_MB] [--low-mem]
                [--use-plugin USE_PLUGIN] [--anat-only] [--boilerplate]
                [--ignore-aroma-denoising-errors] [--error-on-aroma-warnings]
                [-v] [--debug]
                [--ignore {fieldmaps,slicetiming,sbref} [{fieldmaps,slicetiming,sbref} ...]]
                [--longitudinal] [--t2s-coreg]
                [--output-spaces OUTPUT_SPACES [OUTPUT_SPACES ...]]
                [--output-space {T1w,template,fsnative,fsaverage,fsaverage6,fsaverage5} [{T1w
                [--template {MNI152NLin2009cAsym}]
                [--template-resampling-grid TEMPLATE_RESAMPLING_GRID]
                [--bold2t1w-dof {6,9,12}] [--force-bbr] [--force-no-bbr]
                [--medial-surface-nan] [--dummy-scans DUMMY_SCANS]
                [--use-aroma]
                [--aroma-melodic-dimensionality AROMA_MELODIC_DIMENSIONALITY]
                [--return-all-components]
                [--fd-spike-threshold FD_SPIKE_THRESHOLD]
                [--dvars-spike-threshold DVARS_SPIKE_THRESHOLD]
                [--skull-strip-template SKULL_STRIP_TEMPLATE]
                [--skull-strip-fixed-seed] [--fmap-bspline] [--fmap-no-demean]
                [--use-syn-sdc] [--force-syn] [--fs-license-file PATH]
                [--no-submm-recon] [--cifti-output | --fs-no-reconall]
                [-w WORK_DIR] [--resource-monitor] [--reports-only]
                [--run-uuid RUN_UUID] [--write-graph] [--stop-on-first-crash]
                [--notrack] [--sloppy]
                bids_dir output_dir {participant}
```

fMRIprep usage: https://fmriprep.readthedocs.io/en/stable/usage.html

Positional Arguments

`bids_dir`

the root folder of a BIDS valid dataset (sub-XXXXX folders should be found at the top level in this folder).

`output_dir`

the output path for the outcomes of preprocessing and visual reports

`analysis_level`

Possible choices: participant

processing stage to be run, only "participant" in the case of FMRIPREP (see BIDS-Apps specification).

Options for filtering BIDS queries

`--skip_bids_validation, --skip-bids-validation`

assume the input dataset is BIDS compliant and skip the validation

`--participant_label, --participant-label`

a space delimited list of participant identifiers or a single identifier (the sub- prefix can be removed)

`-t, --task-id`

select a specific task to be processed

Options to handle performance

`--nthreads, --n_cpus, -n-cpus`

maximum number of threads across all processes

`--omp-nthreads`

maximum number of threads per-process

`--mem_mb, --mem-mb`

upper bound memory limit for FMRIPREP processes

`--low-mem`

attempt to reduce memory usage (will increase disk usage in working directory)

fMRIprep usage: https://fmriprep.readthedocs.io/en/stable/usage.html

Workflow configuration

`--ignore`

Possible choices: fieldmaps, slicetiming, sbref

ignore selected aspects of the input dataset to disable corresponding parts of the workflow (a space delimited list)

`--longitudinal`

treat dataset as longitudinal - may increase runtime

`--t2s-coreg`

If provided with multi-echo BOLD dataset, create T2*-map and perform T2*-driven coregistration. When multi-echo data is provided and this option is not enabled, standard EPI-T1 coregistration is performed using the middle echo.

`--output-spaces`

Standard and non-standard spaces to resample anatomical and functional images to. Standard spaces may be specified by the form `<TEMPLATE>[:res-<resolution>][:cohort-<label>][...]`, where `<TEMPLATE>` is a keyword (valid keywords: "MNI152Lin", "MNI152NLin2009cAsym", "MNI152NLin6Asym", "MNI152NLin6Sym", "NKI", "OASIS30ANTs", "PNC", "fsLR", "fsaverage") or path pointing to a user-supplied template, and may be followed by optional, colon-separated parameters. Non-standard spaces (valid keywords: anat, T1w, run, func, sbref, fsnative) imply specific orientations and sampling grids

Specific options for running ICA_AROMA

`--use-aroma`

add ICA_AROMA to your preprocessing stream

Specific options for FreeSurfer preprocessing

`--fs-license-file`

Path to FreeSurfer license key file. Get it (for free) by registering at https://surfer.nmr.mgh.harvard.edu/registration.html

Surface preprocessing options

`--no-submm-recon`

disable sub-millimeter (hires) reconstruction

`--cifti-output`

output BOLD files as CIFTI dtseries

`--fs-no-reconall, --no-freesurfer`

disable FreeSurfer surface preprocessing. Note : *–no-freesurfer* is deprecated and will be removed in 1.2. Use *–fs-no-reconall* instead.

Other options

`-w, --work-dir`

path where intermediate results should be stored

`--resource-monitor`

enable Nipype's resource monitoring to keep track of memory and CPU usage

`--reports-only`

only generate reports, don't run workflows. This will only rerun report aggregation, not reportlet generation for specific nodes.

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
#
singularity run --cleanenv \
--home /data/bswift-1/${USER}:/home/${USER} \
--bind /data/bswift-1/${USER}/templateflow:/home/${USER}/templateflow \
/data/bswift-1/jmerch/fmriprep-1.4.1.simg \
/home/jmerch/JAM/in /home/jmerch/JAM/out participant \
--participant-label ${Sub} \
-w /home/jmerch/JAM/out/working \
--skull-strip-template MNI152NLin2009cAsym --output-spaces MNI152NLin2009cAsym:res-2 MNI152NLin6Asym:res-2 T1w fsaverage5 \
--use-aroma \
--nthreads 6 --n_cpus 6 --omp-nthreads 6 \
--mem-mb 24000 \
--cifti-output \
--skip_bids_validation \
--no-submm-recon \
--fs-license-file /home/jmerch/CHT/license.txt
```

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
singularity run --cleanenv \
--bind /data/bswift-1/${USER}/templateflow:/home/
/data/bswift-1/jmerch/fmriprep-1.4.1.simg \
/home/jmerch/JAM/in /home/jmerch/JAM/out partici
--participant-label ${Sub} \
-w /home/jmerch/JAM/out/working \
--skull-strip-template MNI152NLin2009cAsym --output-spaces MNI152NLin2009cAsym:res-2 MNI152NL
--use-aroma \
--nthreads 6 --n_cpus 6 --omp-nthreads 6 \
--mem-mb 24000 \
--cifti-output \
--skip_bids_validation \
--no-submm-recon \
--fs-license-file /home/jmerch/CHT/license.txt
```

Run a singularity container with a clean environment.

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
#
--home /data/bswift-1/${USER}:/home/${USER} \
--bind /data/bswift-1/${USER}/templateflow:/home/${USER}/templateflow \
/home/jmerch/JAM/in /home/jmerch/JAM/out participant \
--participant-label ${Sub} \
-w /home/jmerch/JAM/out/working \
--skull-strip-template MNI152NLin2009cAsym --output-spaces MNI152NL
--use-aroma \
--nthreads 6 --n_cpus 6 --omp-nthreads 6 \
--mem-mb 24000 \
--cifti-output \
--skip_bids_validation \
--no-submm-recon \
--fs-license-file /home/jmerch/CHT/license.txt
```

Bind the local directory that you're working from to the /home directory inside the container. This can get very tricky, so I recommend having everything you need for fMRIprep in one directory on your local environment because singularity wants to run everything from home. Everything on the left side of the colon will be given the new path indicated on the right side of the colon inside the container.

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
#
singularity run --cleanenv \
--home /data/bswift-1/${USER}:/home/${USER} \
/data/bswift-1/jmerch/fmriprep-1.4.1.simg \
--participant-label ${Sub} \
-w /home/jmerch/JAM/out/working \
--skull-strip-template MNI152NLin2009cAsym --output-spaces MNI15
--use-aroma \
--nthreads 6 --n_cpus 6 --omp-nthreads 6 \
--mem-mb 24000 \
--cifti-output \
--skip_bids_validation \
--no-submm-recon \
--fs-license-file /home/jmerch/CHT/license.txt
```

This is the last time you will use your local path, and it is to indicate where the fmriprep singularity image is. These singularity images usually have .simg or .sif

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
#
singularity run --cleanenv \
--home /data/bswift-1/${USER}:/home/${USER} \
--bind /data/bswift-1/${USER}/templateflow:/home/${USER}/templateflow \
home/jmerch/JAM/in /home/jmerch/JAM/out participant \
-w /home/jmerch/JAM/out/working \
--skull-strip-template MNI152NLin2009cAsym --output-spaces
--use-aroma \
--nthreads 6 --n_cpus 6 --omp-nthreads 6 \
--mem-mb 24000 \
--cifti-output \
--skip_bids_validation \
--no-submm-recon \
--fs-license-file /home/jmerch/CHT/license.txt
```

The positional arguments required. Input BIDS directory, output directory where you want ‘prepped data written, and analysis level is participant. NOTE that the paths are the new paths from inside the container!

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
#
singularity run --cleanenv \
--home /data/bswift-1/${USER}:/home/${USER} \
--bind /data/bswift-1/${USER}/templateflow:/home/${USER}/templateflow \
/data/bswift-1/jmerch/fmriprep-1.4.1.simg \
--participant-label ${Sub} \
--skull-strip-template MNI152NLin2009cAsym --output-spaces MNI
--use-aroma \
--nthreads 6 --n_cpus 6 --omp-nthreads 6 \
--mem-mb 24000 \
--cifti-output \
--skip_bids_validation \
--no-submm-recon \
--fs-license-file /home/jmerch/CHT/license.txt
```

T1w fsave

The participant you want to ‘prep

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
singularity run --cleanenv \
--home /data/bswift-1/${USER}:/home/${USER} \
/data/bswift-1/jmerch/fmriprep-1.4.0.simg \
/home/jmerch/JAM/in /home/jmerch/JAM/out participant \
-w /home/jmerch/JAM/out/working \
--use-aroma \
--nthreads 6 --n_cpus 6 --omp-nthrea
--mem-mb 24000 \
--cifti-output \
--resource-monitor \
--skip_bids_validation \
--fs-license-file /home/jmerch/CHT/license.txt
```

Working directory where you want intermediate files saved.

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
#
singularity run --cleanenv \
--home /data/bswift-1/${USER}:/home/${USER} \
--bind /data/bswift-1/${USER}/templateflow:/home/${USER}/templateflow \
/data/bswift-1/jmerch/fmriprep-1.4.1.simg \
/home/jmerch/JAM/in /home/jmerch/JAM/out participant \
--participant-label ${Sub} \
--skull-strip-template MNI152NLin2009cAsym --output-spaces MNI152NLin2009cAsym:res-2 MNI152NLin6Asym:res-2 T1w fsaverage5 \
--nthreads 6 --n_cpus 6 --omp-nthreads 6 \
--mem-mb 24000 \
--cifti-output \
--skip_bids_validation \
--no-submm-recon \
--fs-license-file /home/jmerch/CHT/license.txt
```

Output spaces you want data written to. T1w indicates native space to their T1 scan, template uses the default template. This option is changing, so check usage notes for how to use this for future versions.

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
singularity run --cleanenv \
--home /data/bswift-1/${USER}:/home/${USER} \
/data/bswift-1/jmerch/fmriprep-1.4.0.simg \
/home/jmerch/JAM/in /home/jmerch/JAM/out participant \
--participant-label sub-JAM057 \
-w /home/jmerch/JAM/out/working \
--output-space {T1w template} \
--use-aroma \
--mem-mb 24000 \
--cifti-output \
--resource-monitor \
--skip_bids_validation \
--fs-license-file /home/jmerch/CHT/license.txt
```

Use AROMA (optional)

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
singularity run --cleanenv \
--home /data/bswift-1/${USER}:/home/${USER} \
/data/bswift-1/jmerch/fmriprep-1.4.0
/home/jmerch/JAM/in /home/jmerch/JAM
--participant-label sub-JAM057 \
-w /home/jmerch/JAM/out/working \
--output-space {T1w,template} \
--nthreads 6 --n_cpus 6 --omp-nthreads 12 \
--mem-mb 24000 \
--resource-monitor \
--skip_bids_validation \
--fs-license-file /home/jmerch/CHT/license.txt
```

Hardware specifications based on least powerful nodes on bswift

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
#
singularity run --cleanenv \
--home /data/bswift-1/${USER}:/home/${USER} \
--bind /data/bswift-1/${USER}/templateflow:/home/${US
/data/bswift-1/jmerch/fmriprep-1.4.1.simg \
/home/jmerch/JAM/in /home/jmerch/JAM/out participant
--participant-label ${Sub} \
-w /home/jmerch/JAM/out/working \
--skull-strip-template MNI152NLin2009cAsym --output-s
--use-aroma \
--nthreads 6 --n_cpus 6 --omp-nthreads 6 \
--mem-mb 24000 \
--cifti-output \
--skip_bids_validation \
--no-submm-recon \
--fs-license-file /home/jmerch/CHI/license.txt
```

Cifti output is the human connectome project file format, and great for surface based analyses. I skip bids validation because I don't have the timing files there. Finally, no-submm-recon is recommended to speed up processing. Even freesurfer recommends against it if your T1 is higher resolution than .75 mm

Our fMRIprep usage via Singularity (when it gets a little tricky)

```
singularity run --cleanenv \
--home /data/bswift-1/${USER}:/home/
/data/bswift-1/jmerch/fmriprep-1.4.0
/home/jmerch/JAM/in /home/jmerch/JAM
--participant-label sub-JAM057 \
-w /home/jmerch/JAM/out/working \
--output-space {T1w,template} \
--use-aroma \
--nthreads 6 --n_cpus 6 --omp-nthrea
--mem-mb 24000 \
--cifti-output \
--resource-monitor \
--skip_bids_validation \
--fs-license-file /home/jmerch/CHT/license.txt
```

This is the last where you need to specify the path to the freesurfer license using the within container path. You can download a FS license for free from their website.

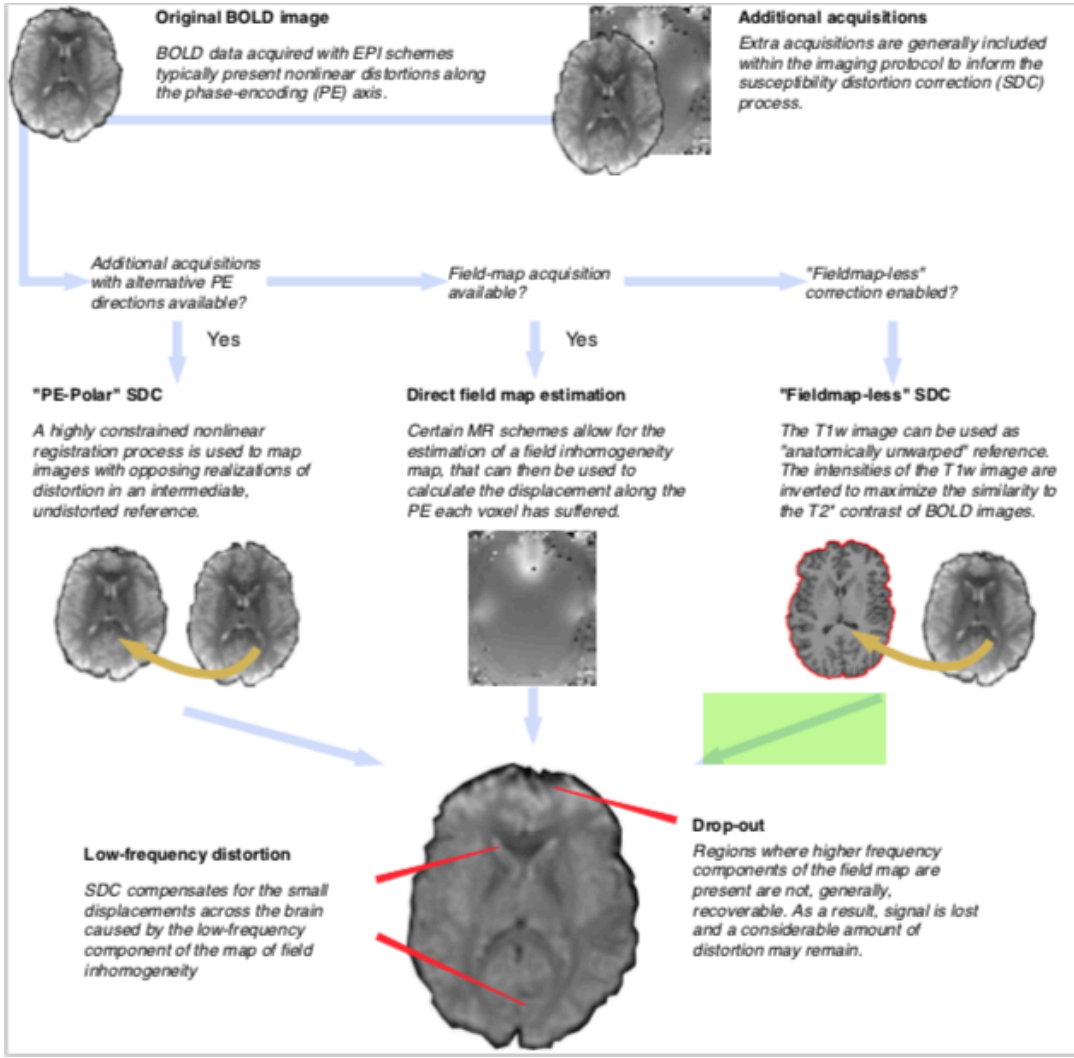
Putting it all together for a SLURM batch script

You can reference the SLURM presentation in the shared google folder for more detailed explanation of the slurm options.

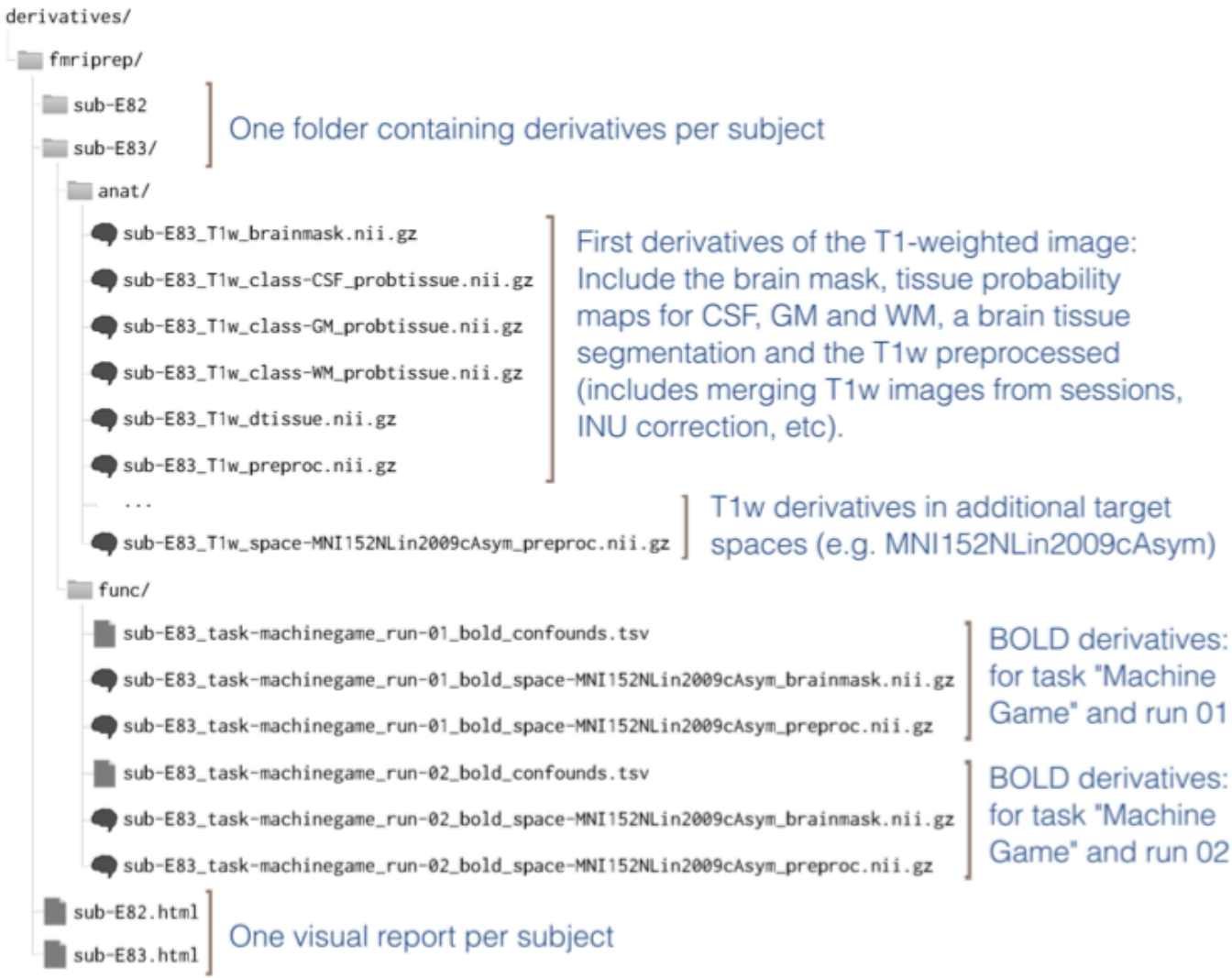
```
#!/bin/bash
#SBATCH --time=168:00:00
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=6
#SBATCH --mem=24000
#SBATCH --output=JAM057.log
#SBATCH --mail-user=jmerch@terpmail.umd.edu
#SBATCH --mail-type=ALL
#
# load the singularity module
module load singularity/2.6.0
#
#
# You can change the 4 lines below, I just like having it time stamp it
echo "-----------------------------------------------------------------"
echo "Starting fMRIprep at:"
echo "working on sub-JAM057"
date
echo "-----------------------------------------------------------------"
#
#
#
singularity run --cleanenv \
--home /data/bswift-1/${USER}:/home/${USER} \
/data/bswift-1/jmerch/fmriprep-1.4.0.simg \
/home/jmerch/JAM/in /home/jmerch/JAM/out participant \
--participant-label sub-JAM057 \
-w /home/jmerch/JAM/out/working \
--output-space {T1w,template} \
--use-aroma \
--nthreads 6 --n_cpus 6 --omp-nthreads 12 \
--mem-mb 24000 \
--cifti-output \
--resource-monitor \
--skip_bids_validation \
--fs-license-file /home/jmerch/CHT/license.txt
#
echo "-----------------------------------------------------------------"
echo "Ended fMRIprep"
echo "on sub-JAM057"
date
echo "-----------------------------------------------------------------"
```

Note on fieldmaps

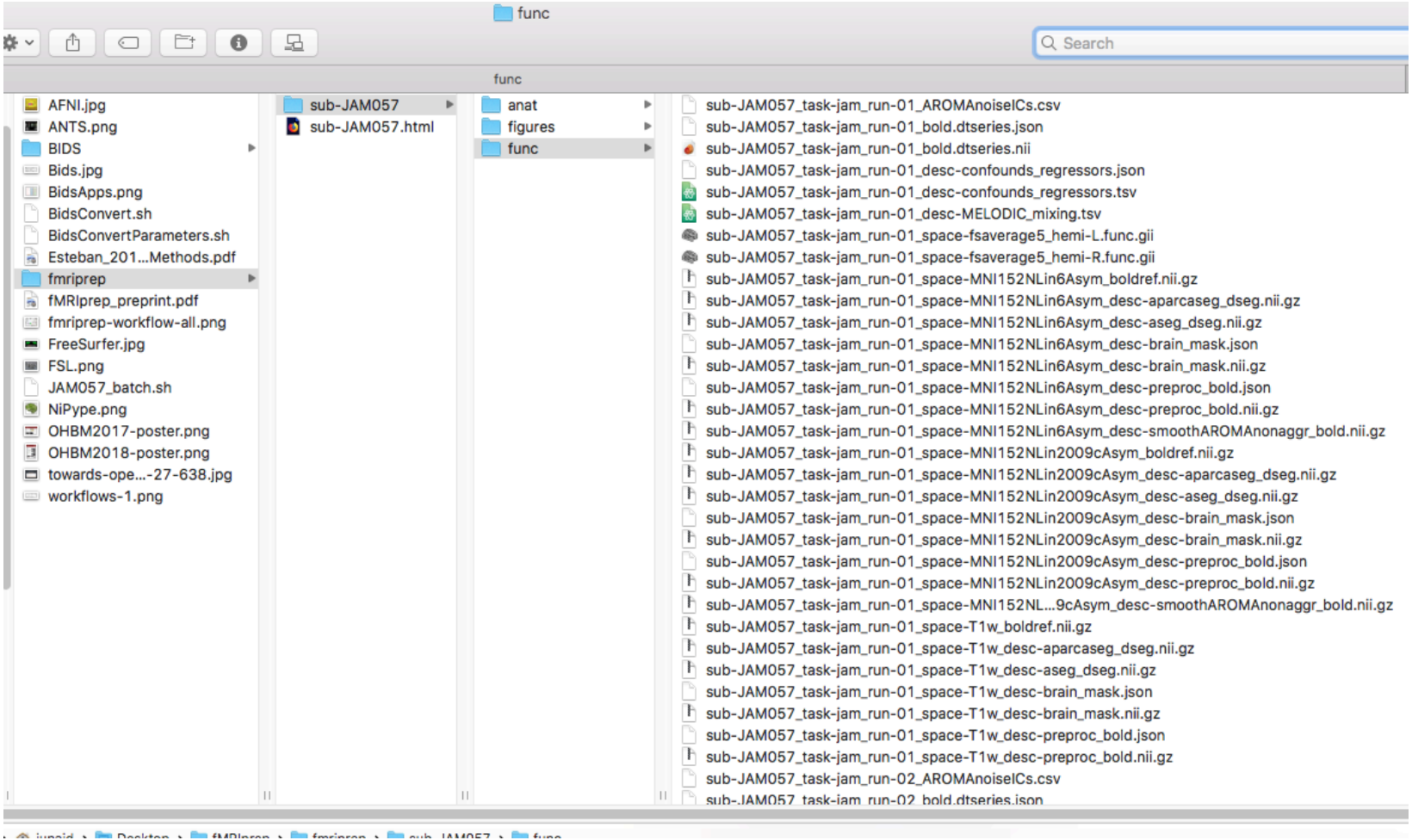
I used a really hacky sed command approach for editing the .json files of our fmap scans so that fmriprep would use them for fieldmap correction. It's not great, and I would like to improve it, but it works for now.



fMRIprep outputs: Folder



fMRIprep outputs: Folder



Naming conventions (similar to BIDS):

All file names start first with the sub-<subjectid>, followed by an underscore. E.g. **sub-JAM057_**

anat/:

‘desc’ indicates native space

‘preproc’ indicates preprocessed (INU in this case)

‘from’ files are text files with warping parameters to different spaces

.gii files are freesurfer things

‘space-’ files indicate what space they’re in

sub-JAM057_space-MNI152NLin2009cAsym_desc-preproc_T1w.nii.gz is normalized structural to the MNI152NLin2009 Asymetric space.

```
sub-JAM057_desc-aparcaseg_dseg.nii.gz
sub-JAM057_desc-aseg_dseg.nii.gz
sub-JAM057_desc-brain_mask.json
sub-JAM057_desc-brain_mask.nii.gz
sub-JAM057_desc-preproc_T1w.json
sub-JAM057_desc-preproc_T1w.nii.gz
sub-JAM057_dseg.nii.gz
sub-JAM057_from-MNI152NLin2009cAsym_to-T1w_mode-image_xfm.h5
sub-JAM057_from-MNI152NLin6Asym_to-T1w_mode-image_xfm.h5
sub-JAM057_from-T1w_to-MNI152NLin2009cAsym_mode-image_xfm.h5
sub-JAM057_from-T1w_to-MNI152NLin6Asym_mode-image_xfm.h5
sub-JAM057_from-T1w_to-fsnative_mode-image_xfm.txt
sub-JAM057_from-orig_to-T1w_mode-image_xfm.txt
sub-JAM057_hemi-L_inflated.surf.gii
sub-JAM057_hemi-L_midthickness.surf.gii
sub-JAM057_hemi-L_pial.surf.gii
sub-JAM057_hemi-L_smoothwm.surf.gii
sub-JAM057_hemi-R_inflated.surf.gii
sub-JAM057_hemi-R_midthickness.surf.gii
sub-JAM057_hemi-R_pial.surf.gii
sub-JAM057_hemi-R_smoothwm.surf.gii
sub-JAM057_label-CSF_probseg.nii.gz
sub-JAM057_label-GM_probseg.nii.gz
sub-JAM057_label-WM_probseg.nii.gz
sub-JAM057_space-MNI152NLin2009cAsym_desc-brain_mask.json
sub-JAM057_space-MNI152NLin2009cAsym_desc-brain_mask.nii.gz
sub-JAM057_space-MNI152NLin2009cAsym_desc-preproc_T1w.json
sub-JAM057_space-MNI152NLin2009cAsym_desc-preproc_T1w.nii.gz
sub-JAM057_space-MNI152NLin2009cAsym_dseg.nii.gz
sub-JAM057_space-MNI152NLin2009cAsym_label-CSF_probseg.nii.gz
sub-JAM057_space-MNI152NLin2009cAsym_label-GM_probseg.nii.gz
sub-JAM057_space-MNI152NLin2009cAsym_label-WM_probseg.nii.gz
sub-JAM057_space-MNI152NLin6Asym_desc-brain_mask.json
sub-JAM057_space-MNI152NLin6Asym_desc-brain_mask.nii.gz
sub-JAM057_space-MNI152NLin6Asym_desc-preproc_T1w.json
sub-JAM057_space-MNI152NLin6Asym_desc-preproc_T1w.nii.gz
sub-JAM057_space-MNI152NLin6Asym_dseg.nii.gz
sub-JAM057_space-MNI152NLin6Asym_label-CSF_probseg.nii.gz
sub-JAM057_space-MNI152NLin6Asym_label-GM_probseg.nii.gz
sub-JAM057_space-MNI152NLin6Asym_label-WM_probseg.nii.gz
```

Naming conventions (similar to BIDS):

func/:
AROMAnoiseICs is a csv of the components

dtseries is cifti output

MELODIC is melodic ICA output (?) & the confounds regressor is massive list of potential regressors you can use for first level, like motion, FD, CompCor, and much more!

_boldref.nii.gz is reference single image

space-<template>_desc-preproc_bold.nii.gz are the files that are fully preprocessed in the space listed in the template position 4d file

smoothAROMAnonaggr is the AROMA denoised image

_space-T1w is native space

```
sub-JAM057_task-jam_run-01_AROMAnoiseICs.csv
sub-JAM057_task-jam_run-01_bold.dtseries.json
sub-JAM057_task-jam_run-01_bold.dtseries.nii
sub-JAM057_task-jam_run-01_desc-MELODIC_mixing.tsv
sub-JAM057_task-jam_run-01_desc-confounds_regressors.json
sub-JAM057_task-jam_run-01_desc-confounds_regressors.tsv
sub-JAM057_task-jam_run-01_space-MNI152NLin2009cAsym_boldref.nii.gz
sub-JAM057_task-jam_run-01_space-MNI152NLin2009cAsym_desc-aparcaseg_dseg.nii.gz
sub-JAM057_task-jam_run-01_space-MNI152NLin2009cAsym_desc-aseg_dseg.nii.gz
sub-JAM057_task-jam_run-01_space-MNI152NLin2009cAsym_desc-brain_mask.json
sub-JAM057_task-jam_run-01_space-MNI152NLin2009cAsym_desc-brain_mask.nii.gz
sub-JAM057_task-jam_run-01_space-MNI152NLin2009cAsym_desc-preproc_bold.json
sub-JAM057_task-jam_run-01_space-MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
sub-JAM057_task-jam_run-01_space-MNI152NLin2009cAsym_desc-smoothAROMAnonaggr_bold.nii.gz
sub-JAM057_task-jam_run-01_space-MNI152NLin6Asym_boldref.nii.gz
sub-JAM057_task-jam_run-01_space-MNI152NLin6Asym_desc-aparcaseg_dseg.nii.gz
sub-JAM057_task-jam_run-01_space-MNI152NLin6Asym_desc-aseg_dseg.nii.gz
sub-JAM057_task-jam_run-01_space-MNI152NLin6Asym_desc-brain_mask.json
sub-JAM057_task-jam_run-01_space-MNI152NLin6Asym_desc-brain_mask.nii.gz
sub-JAM057_task-jam_run-01_space-MNI152NLin6Asym_desc-preproc_bold.json
sub-JAM057_task-jam_run-01_space-MNI152NLin6Asym_desc-preproc_bold.nii.gz
sub-JAM057_task-jam_run-01_space-MNI152NLin6Asym_desc-smoothAROMAnonaggr_bold.nii.gz
sub-JAM057_task-jam_run-01_space-T1w_boldref.nii.gz
sub-JAM057_task-jam_run-01_space-T1w_desc-aparcaseg_dseg.nii.gz
sub-JAM057_task-jam_run-01_space-T1w_desc-aseg_dseg.nii.gz
sub-JAM057_task-jam_run-01_space-T1w_desc-brain_mask.json
sub-JAM057_task-jam_run-01_space-T1w_desc-brain_mask.nii.gz
sub-JAM057_task-jam_run-01_space-T1w_desc-preproc_bold.json
sub-JAM057_task-jam_run-01_space-T1w_desc-preproc_bold.nii.gz
sub-JAM057_task-jam_run-01_space-fsaverage5_hemi-L.func.gii
sub-JAM057_task-jam_run-01_space-fsaverage5_hemi-R.func.gii
```

Full documentation on Outputs: https://fmriprep.readthedocs.io/en/stable/outputs.html

Outputs of fMRIPrep

FMRIPrep generates three broad classes of outcomes:

1. **Visual QA (quality assessment) reports**: one HTML per subject, that allows the user a thorough visual assessment of the quality of processing and ensures the transparency of fMRIPrep operation.
2. **Pre-processed imaging data** which are derivatives of the original anatomical and functional images after various preparation procedures have been applied. For example, INU-corrected versions of the T1-weighted image (per subject), the brain mask, or BOLD images after head-motion correction, slice-timing correction and aligned into the same-subject's T1w space or into MNI space.
3. **Additional data for subsequent analysis**, for instance the transformations between different spaces or the estimated confounds.

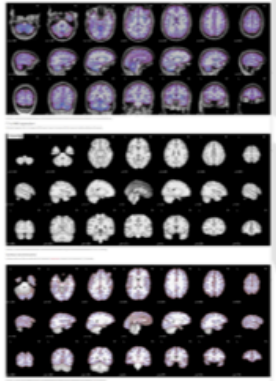
fMRIprep outputs: Report

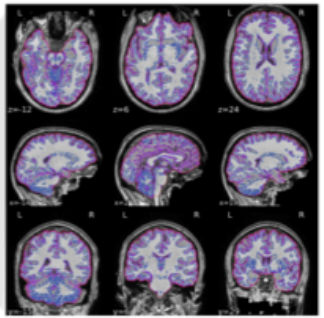
Summary

Reports start with an overview of the dataset, as identified using BIDS.

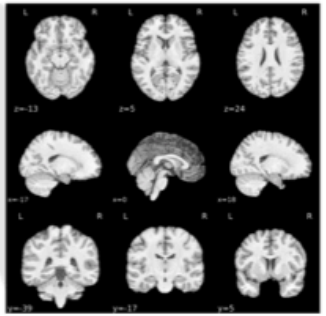
Anatomical processing

Several panels allow for quality control of the anatomical workflow. Brain tissue segmentation, spatial normalization and surface reconstruction (if requested) can be inspected using these visualization panels.





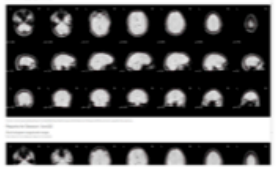
T1-weighted reference, brain mask, intensity inhomogeneity and brain tissue segmentation panel. A static mosaic allows the assessment of these four crucial steps of pre-processing anatomical images.

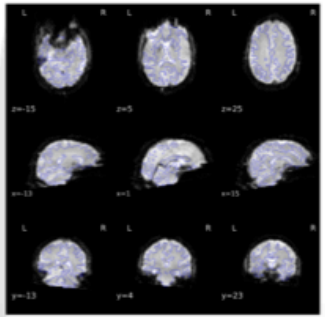


Spatial normalization. A dynamic mosaic that transitions between the target atlas space and the T1w-reference aligned into that space allows checking the accuracy of this image registration process.

Fieldmaps processing

When the dataset contains any of the supported alternatives to estimate the deformation map corresponding to susceptibility distortions, these panels help assess these images were correctly processed.





Susceptibility distortion correction. If fieldmap information was found or the "fieldmap-less" correction is requested, the step is assessed with a dynamic mosaic that transitions between the unwarped ("after") and original ("before"). Contours of the white-matter are also presented as anatomical cue.

Functional processing

Each BOLD run across the different tasks and sessions will be presented at different quality control

fMRIprep outputs: Report

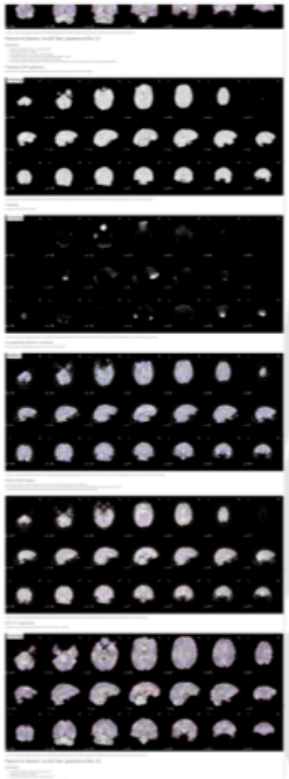
Functional processing

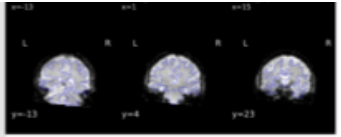
Each BOLD run across the different tasks and sessions will be presented at different quality control points.

First, when fieldmaps were found, some mosaics will show the alignment of those maps to the BOLD reference. The block ends with a dynamic plot showing how images are unwarped.

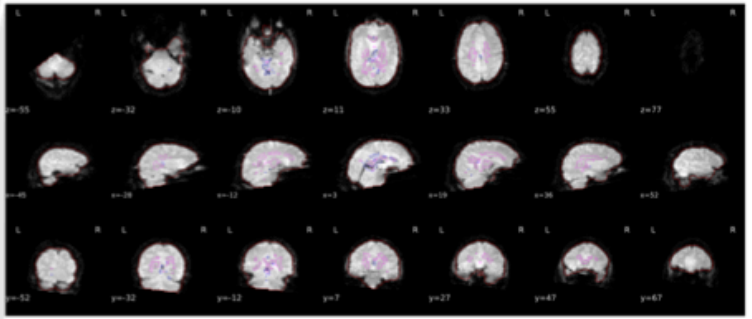
The report also shows processing in native BOLD space plotting the brain mask calculated from the functional MR signal and the regions-of-interest (ROIs) where the CompCor confounds are calculated.

Finally, the alignment between same-subject T1-weighted and that specific BOLD run is presented.

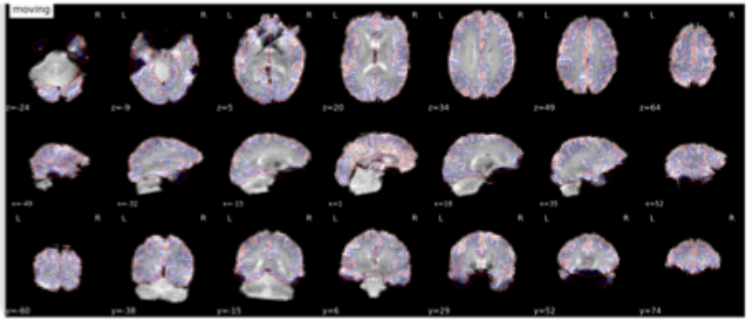




unwarped ("after") and original ("before"). Contours of the white-matter are also presented as anatomical cue.



BOLD mask and CompCor ROIs. The final BOLD signal is presented, with contours representing the outline of the brain mask, and two regions-of-interest (ROIs) where CompCor confounds are estimated.



Alignment of BOLD and the T1w reference. The correct alignment to the anatomical reference is assessed with a dynamic mosaic that renders the reconstructed surfaces over the BOLD reference.

Errors

FMRIPrep is explicit about errors, and any problems encountered along the processing will be listed at the end of the report, with collapsible panels containing the specific detail of each error.

Challenge 2 – fMRIprep

A) Download the fmriprep_reports folder and launch the html file contained therein. How many functional tasks were processed, and how many runs per task? How many spaces are the data processed to (i.e. what templates)? How do you get to the boilerplate methods section that fMRIprep generates?

B) Download the fmriprep_job_script_example.sh script. Read through the script and get an understanding of the options involved in running fmriprep. Now pretend you are running fmriprep on your laptop (I highly recommend against actually running it on your laptop because it will eat up all of your hardware resources and you won't be able to do anything else). Edit this code so that you could run this on your laptop. Pay special attention to the paths because the local paths need to be mapped to the paths within the container.

C) BONUS: If you have a server or powerful workstation that you can access, try running fmriprep using the script you just hacked. You can use the fmriprep-singularity container I provided a link for at the top of this README.