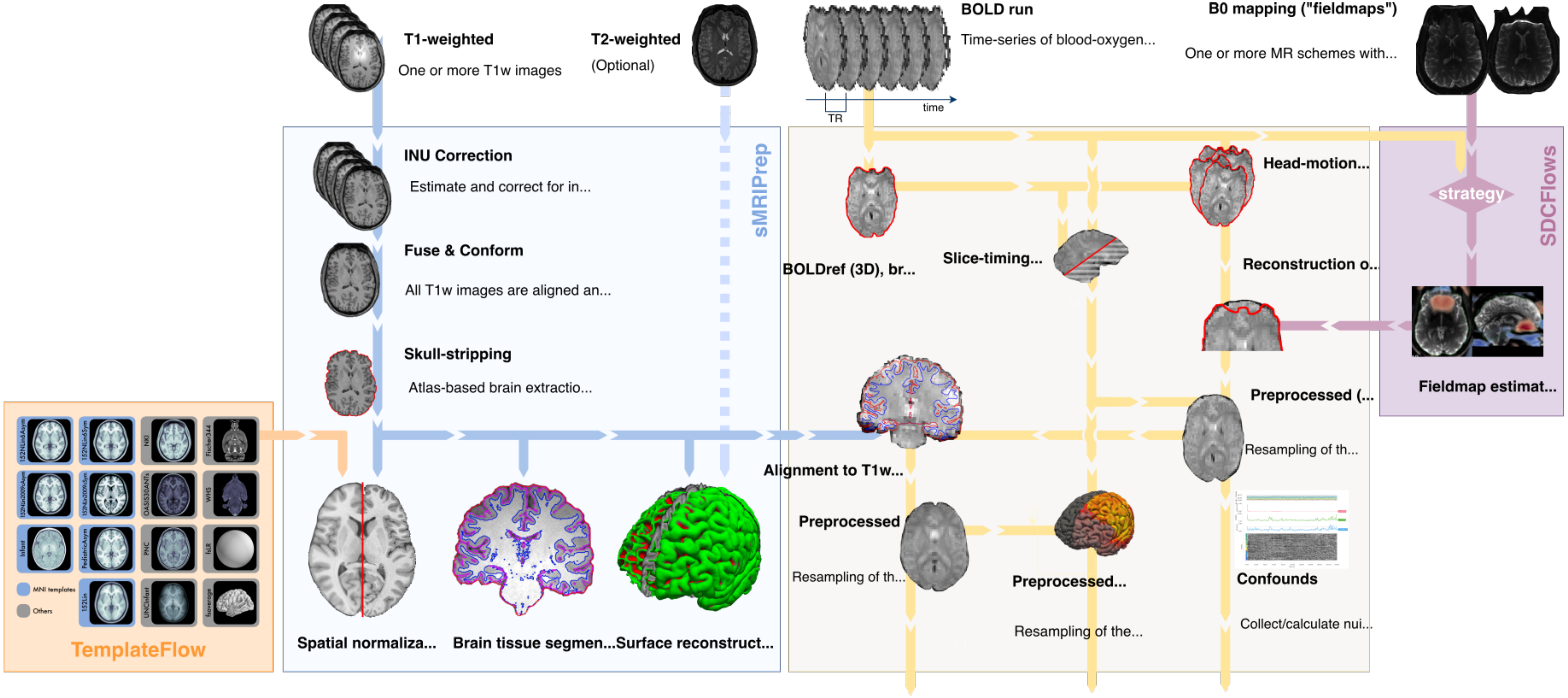
Preprocessing fMRI data using BIDS and fMRIprep

Neal Morton

Preprocessing fMRI data



- Preprocessing tools
- BIDS format
- fMRIprep

Preprocessing tools

Preprocessing tools

Limitations

- Many individual tools: AFNI, SPM, FSL, FreeSurfer, ANTS...
- Installation is complicated and idiosyncratic
- The interface and language used for each package varies
- For optimal preprocessing, need to combine tools that were not designed to work together (e.g., FSL and ANTS)

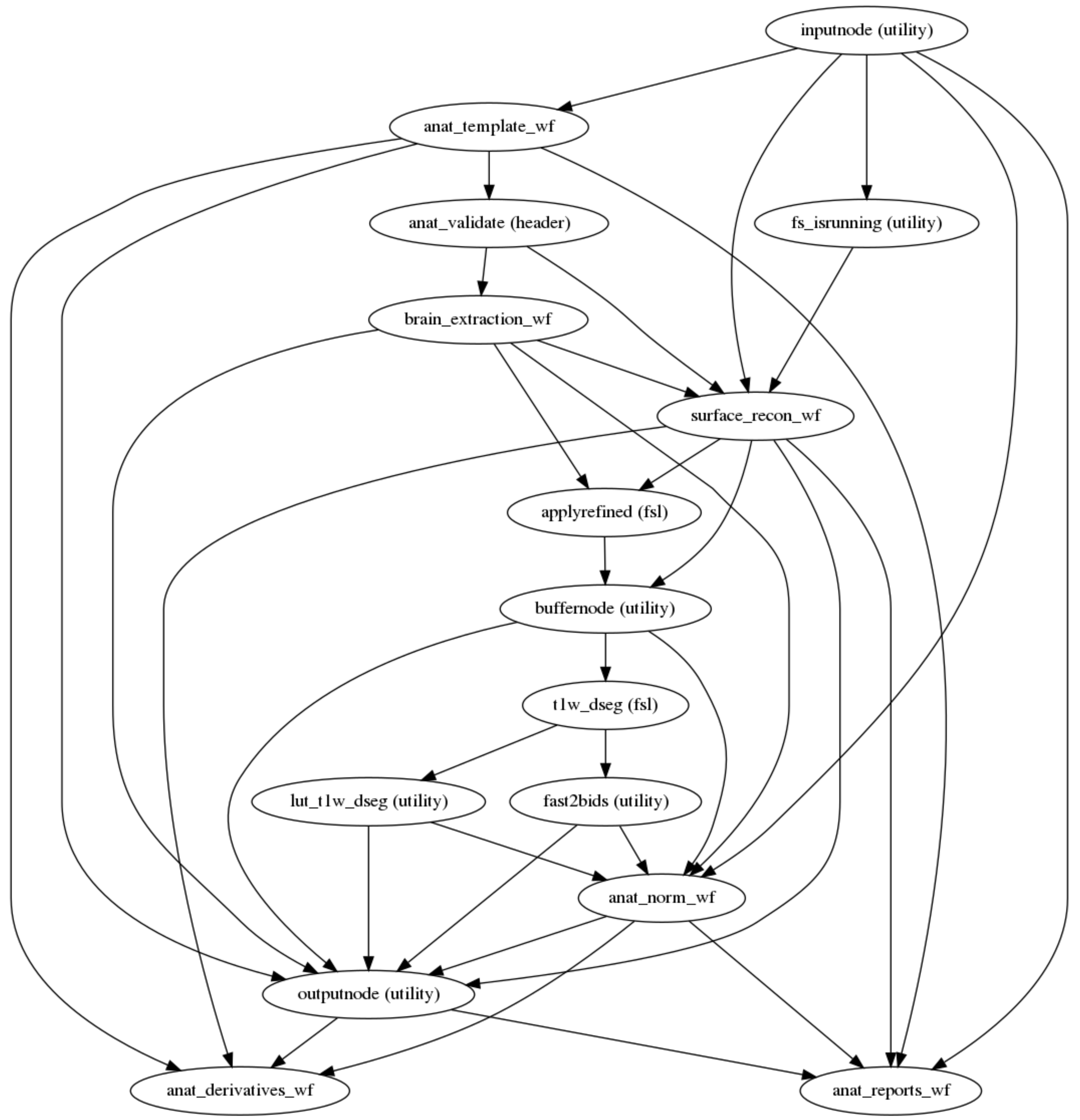
Preprocessing tools

Improvements

- Interoperability made possible by the NIfTI format (toolbox-specific formats remain in FreeSurfer and AFNI, however)
- To further help bridge tools, Nipype was developed
- Docker containers have further helped ease use of multiple software packages together

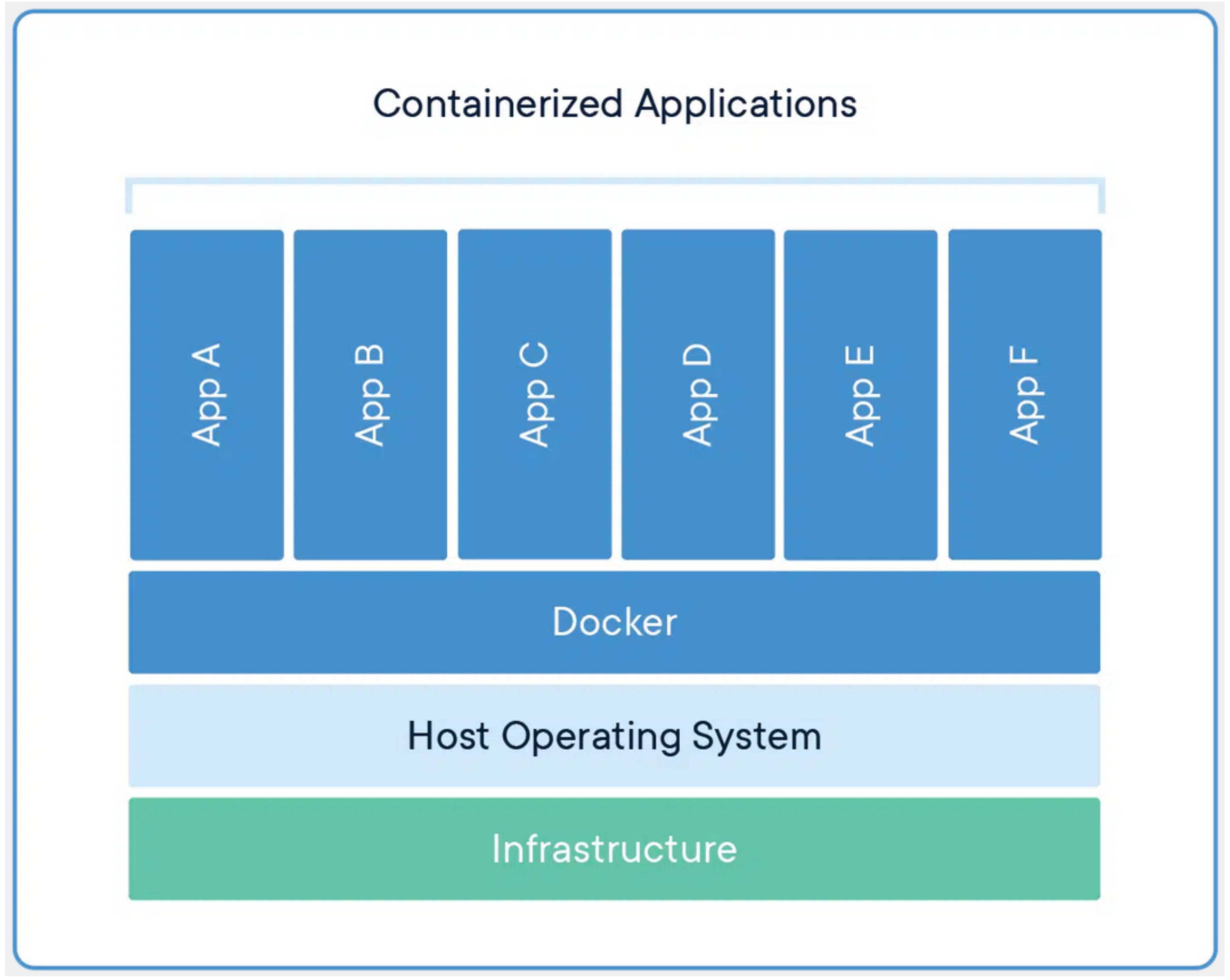
Nipype

- Describes "workflows" as a graph of inputs, operations, and outputs
- A replacement for shell scripts
- Distributes processes over available cores and manages memory usage
- Workflows are available and under active development for multiple modalities
  - Spatial preprocessing
  - Susceptibility distortion correction
  - fMRIprep



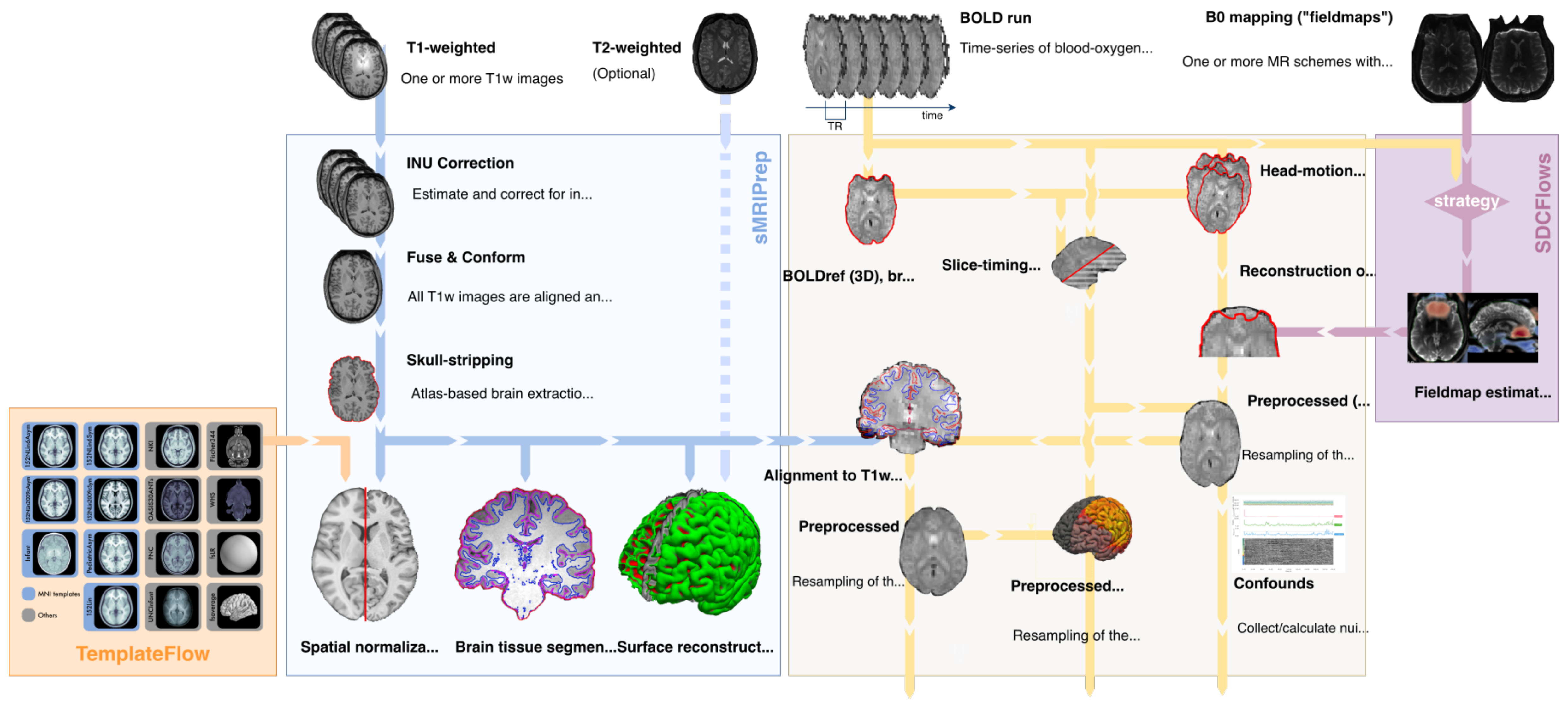
Docker containers

- Containers allow apps, including their dependencies (e.g., FSL, ANTs), to be easily installed and run
- Containers are created from a Dockerfile that specifies how to install everything needed to run the app
- Containers can be downloaded from Docker Hub
- Docker requires root permissions, which doesn't work on a cluster; enter Singularity
- Singularity works on TACC resources, and a number of containers are installed to run software used in the lab



fMRIprep

- Uses Nipype to implement "minimal preprocessing" of fMRI datasets
- Includes anatomical segmentation, slice-timing correction, motion correction, distortion correction, and registration to template space
- Idea is to run a common set of preprocessing for any analysis (univariate, multivariate, connectivity, etc.)
- Uses many different tools under the hood, which are all packaged in a Docker container
- Requires data to be in BIDS format



Preprocessing tools

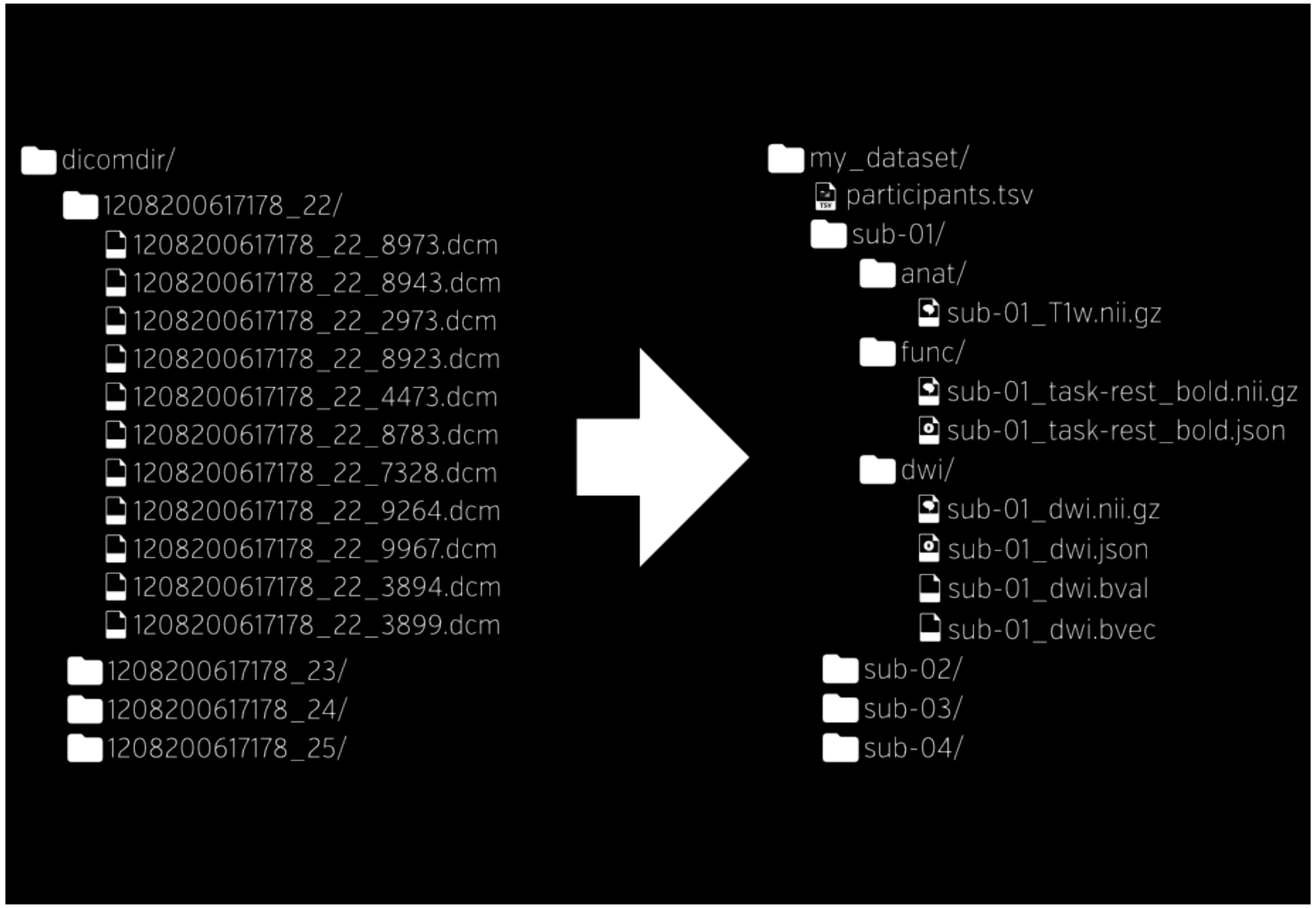
- See the lab wiki for links to documentation for each of these tools
  - https://github.com/prestonlab/wiki/wiki/Running-basic-fMRI-preprocessing-using-BIDS-and-fMRIprep

BIDS format

BIDS

Brain imaging data structure

- Can streamline software development and improve usability with a standard data structure
- Put your data into BIDS format once, then easily run any "BIDS app"
- Data are stored in standard formats (NIfTI, TSV), while metadata are stored in "sidecar" JSON files
- Encourages complete documentation, including scanning parameters and behavioral task events
- Most metadata are extracted automatically

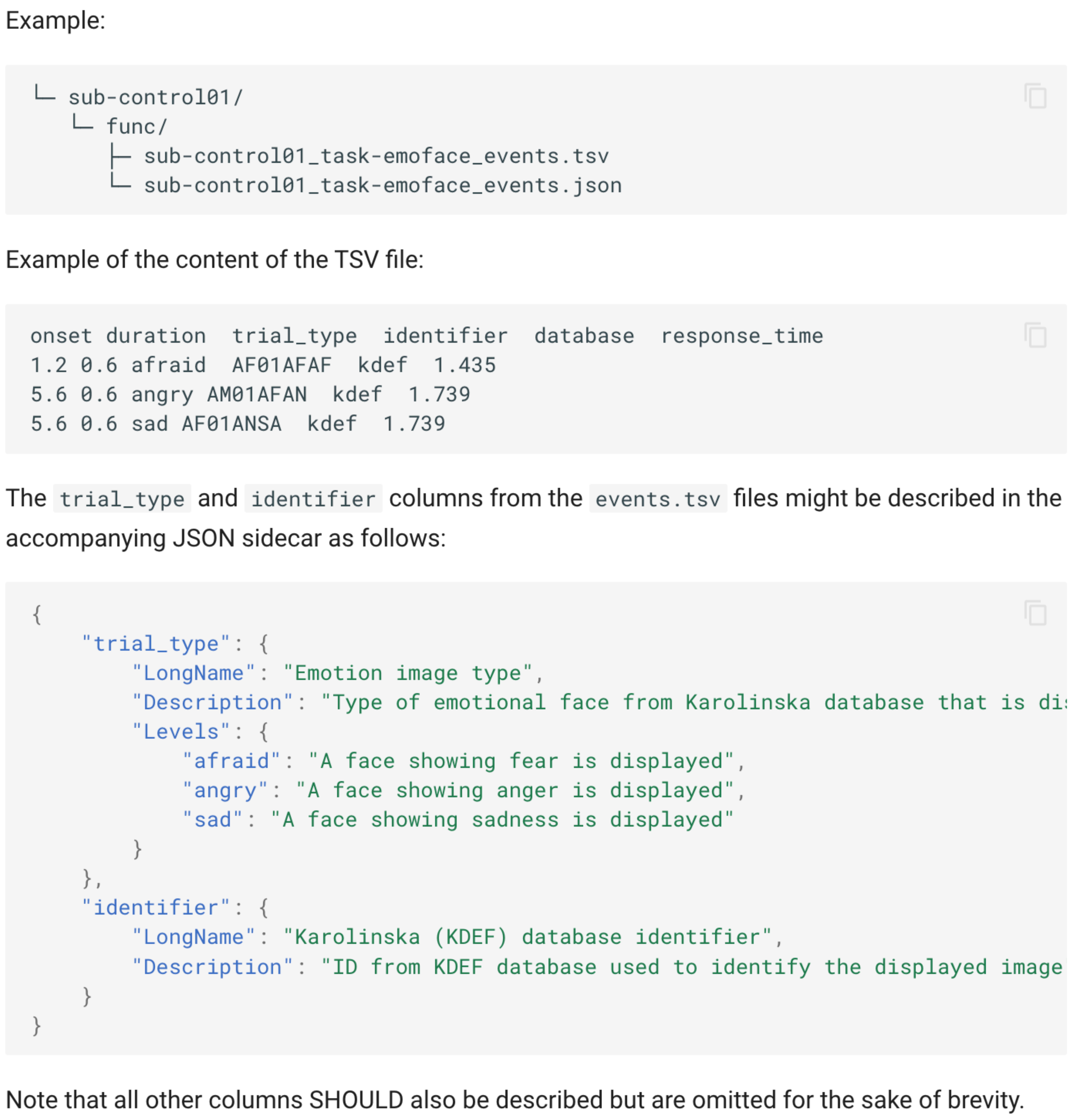


```
dicomdir/
    1208200617178_22/
        1208200617178_22_8973.dcm
        1208200617178_22_8943.dcm
        1208200617178_22_2973.dcm
        1208200617178_22_8923.dcm
        1208200617178_22_4473.dcm
        1208200617178_22_8783.dcm
        1208200617178_22_7328.dcm
        1208200617178_22_9264.dcm
        1208200617178_22_9967.dcm
        1208200617178_22_3894.dcm
        1208200617178_22_3899.dcm
    1208200617178_23/
    1208200617178_24/
    1208200617178_25/
```

```
my_dataset/
    participants.tsv
    sub-01/
        anat/
            sub-01_T1w.nii.gz
        func/
            sub-01_task-rest_bold.nii.gz
            sub-01_task-rest_bold.json
        dwi/
            sub-01_dwi.nii.gz
            sub-01_dwi.json
            sub-01_dwi.bval
            sub-01_dwi.bvec
    sub-02/
    sub-03/
    sub-04/
```

Converting task data

- The BIDS specification includes a simple, flexible format for describing experiment events
- TSV (tab-separated value) file for each run. Can write using any language (Python, R, Matlab, all have good tools for working with tabular data)
- Generally should define onset, duration, and trial_type fields. Add any other fields you need for neural or behavioral analysis
- Fields are described in a JSON sidecar file (can just have one per task, in the main directory of the BIDS structure)
- Can write directly to events format when collecting data, or write a script to create events files after the fact (see Tesser and Bender for examples)



Example:

```
└─ sub-control01/
   └─ func/
      ├─ sub-control01_task-emoface_events.tsv
      └─ sub-control01_task-emoface_events.json
```

Example of the content of the TSV file:

```
onset duration  trial_type  identifier  database  response_time
1.2 0.6 afraid  AF01AFAF  kdef  1.435
5.6 0.6 angry AM01AFAN  kdef  1.739
5.6 0.6 sad AF01ANSA  kdef  1.739
```

The `trial_type` and `identifier` columns from the `events.tsv` files might be described in the accompanying JSON sidecar as follows:

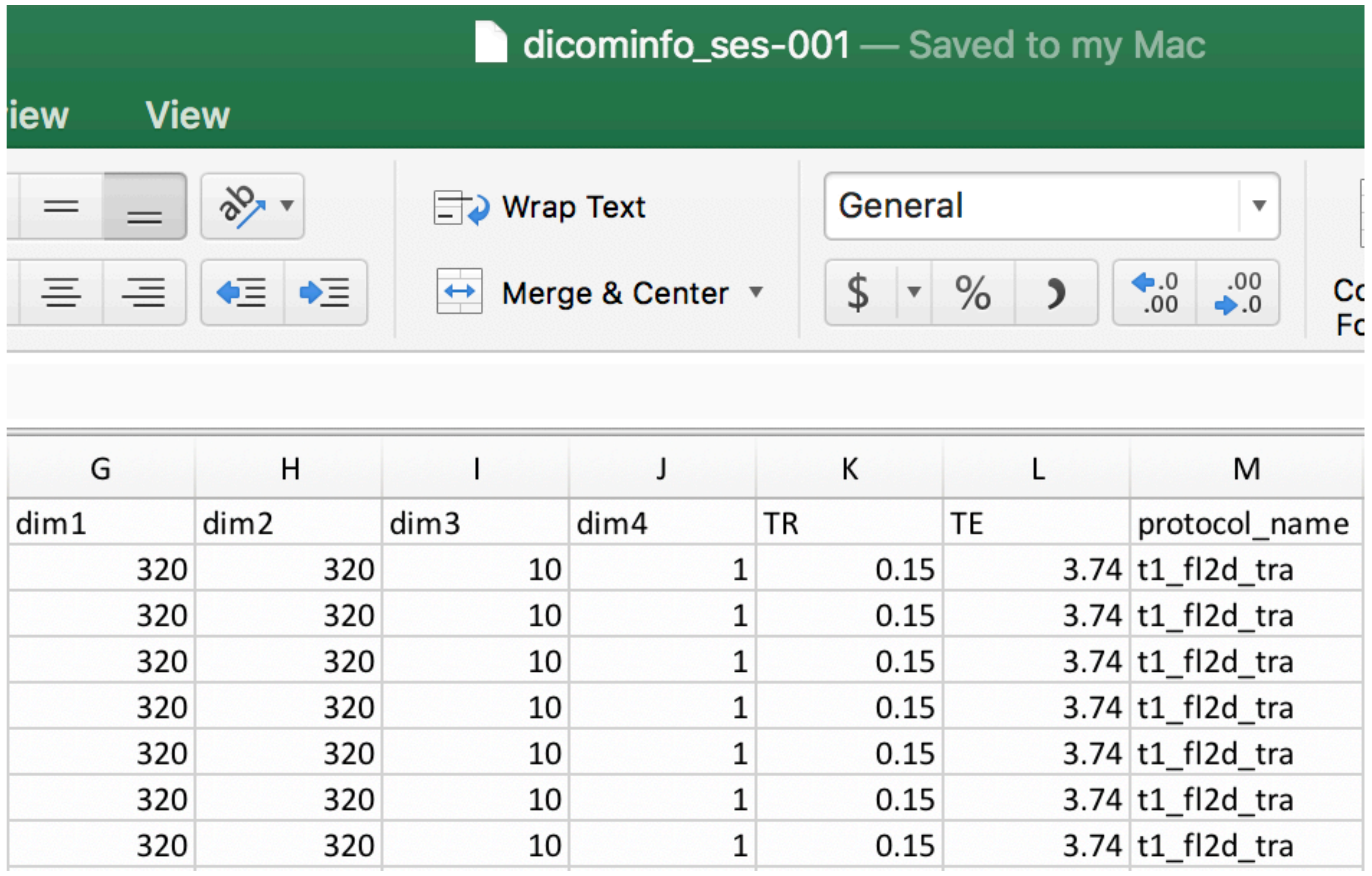
```
{
    "trial_type": {
        "LongName": "Emotion image type",
        "Description": "Type of emotional face from Karolinska database that is di
        "Levels": {
            "afraid": "A face showing fear is displayed",
            "angry": "A face showing anger is displayed",
            "sad": "A face showing sadness is displayed"
        }
    },
    "identifier": {
        "LongName": "Karolinska (KDEF) database identifier",
        "Description": "ID from KDEF database used to identify the displayed image
    }
}
```

Note that all other columns SHOULD also be described but are omitted for the sake of brevity.

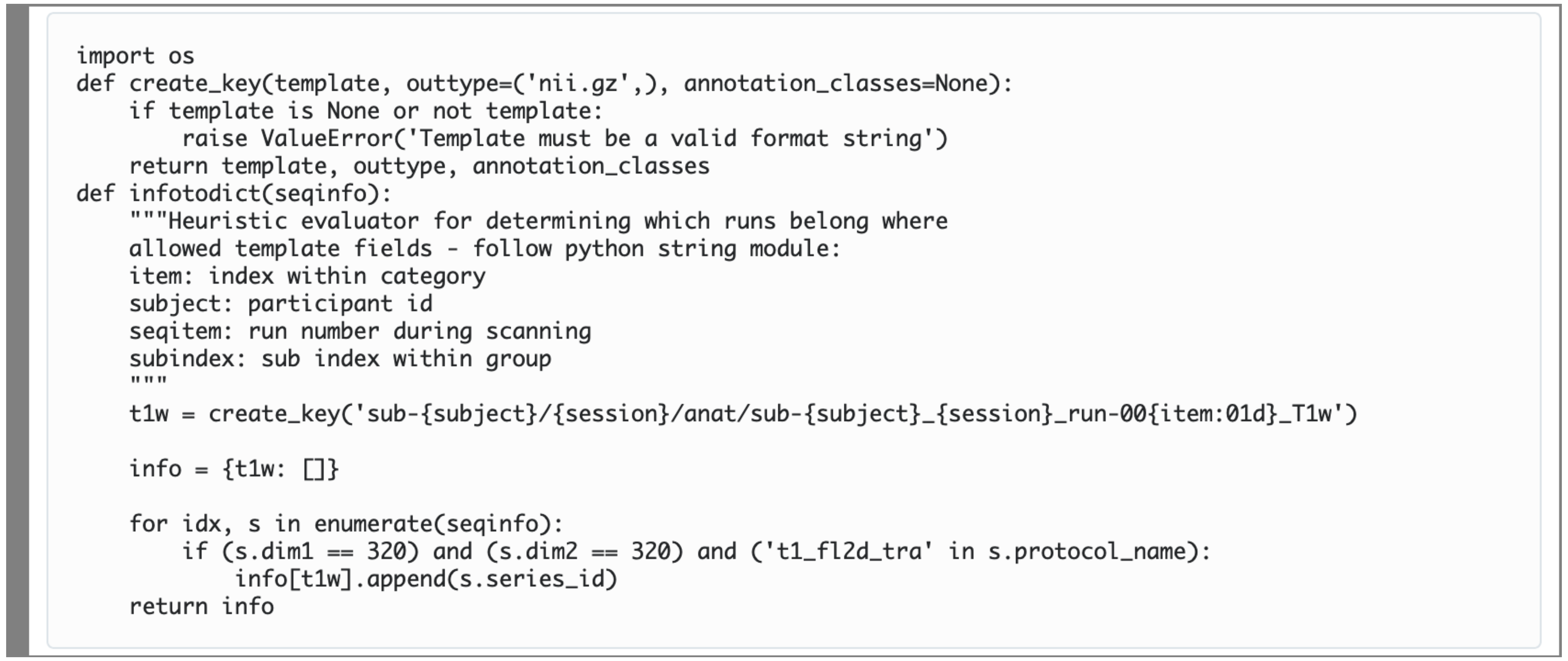
HeuDiConv

Heuristic Dicom Conversion

- HeuDiConv can be configured to convert basically any fMRI dataset to BIDS format
- First, extract information from the DICOM files to get features of each scan
- For each dataset, need to write a Python "heuristic file" that uses those features to put each scan in the correct place
- Once the heuristic is determined, can use HeuDiConv to run the dcm2niix program to convert to NIfTI format
- The main HeuDiConv page doesn't really explain how to use it in practical terms
  - See the Stanford tutorial for a walkthrough



G	H	I	J	K	L	M
dim1	dim2	dim3	dim4	TR	TE	protocol_name
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra

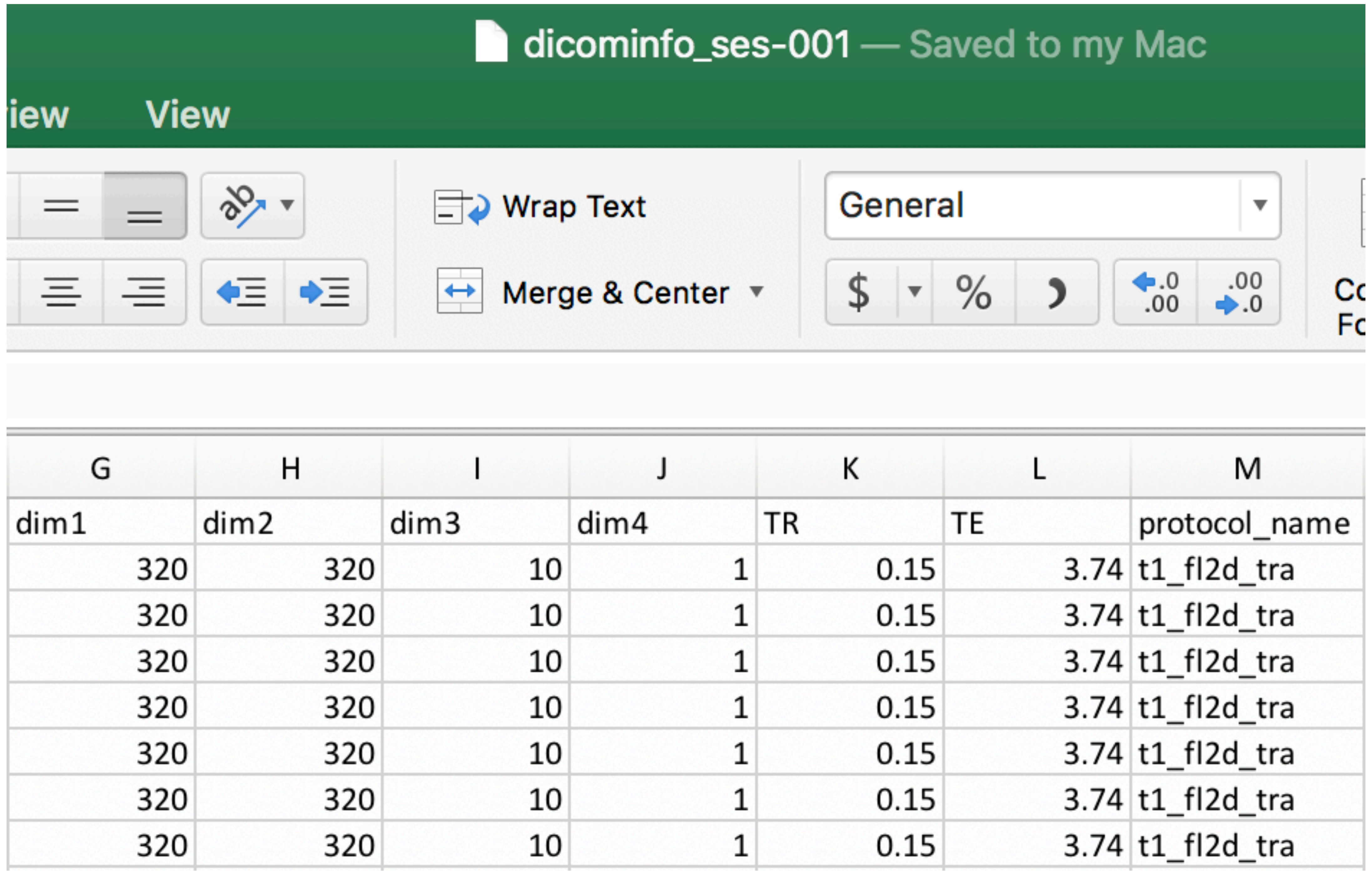


```
import os
def create_key(template, outtype=('nii.gz',), annotation_classes=None):
    if template is None or not template:
        raise ValueError('Template must be a valid format string')
    return template, outtype, annotation_classes
def infotodict(seqinfo):
    """Heuristic evaluator for determining which runs belong where
    allowed template fields - follow python string module:
    item: index within category
    subject: participant id
    seqitem: run number during scanning
    subindex: sub index within group
    """
    t1w = create_key('sub-{subject}/{session}/anat/sub-{subject}_{session}_run-00{item:01d}_T1w')

    info = {t1w: []}

    for idx, s in enumerate(seqinfo):
        if (s.dim1 == 320) and (s.dim2 == 320) and ('t1_fl2d_tra' in s.protocol_name):
            info[t1w].append(s.series_id)
    return info
```

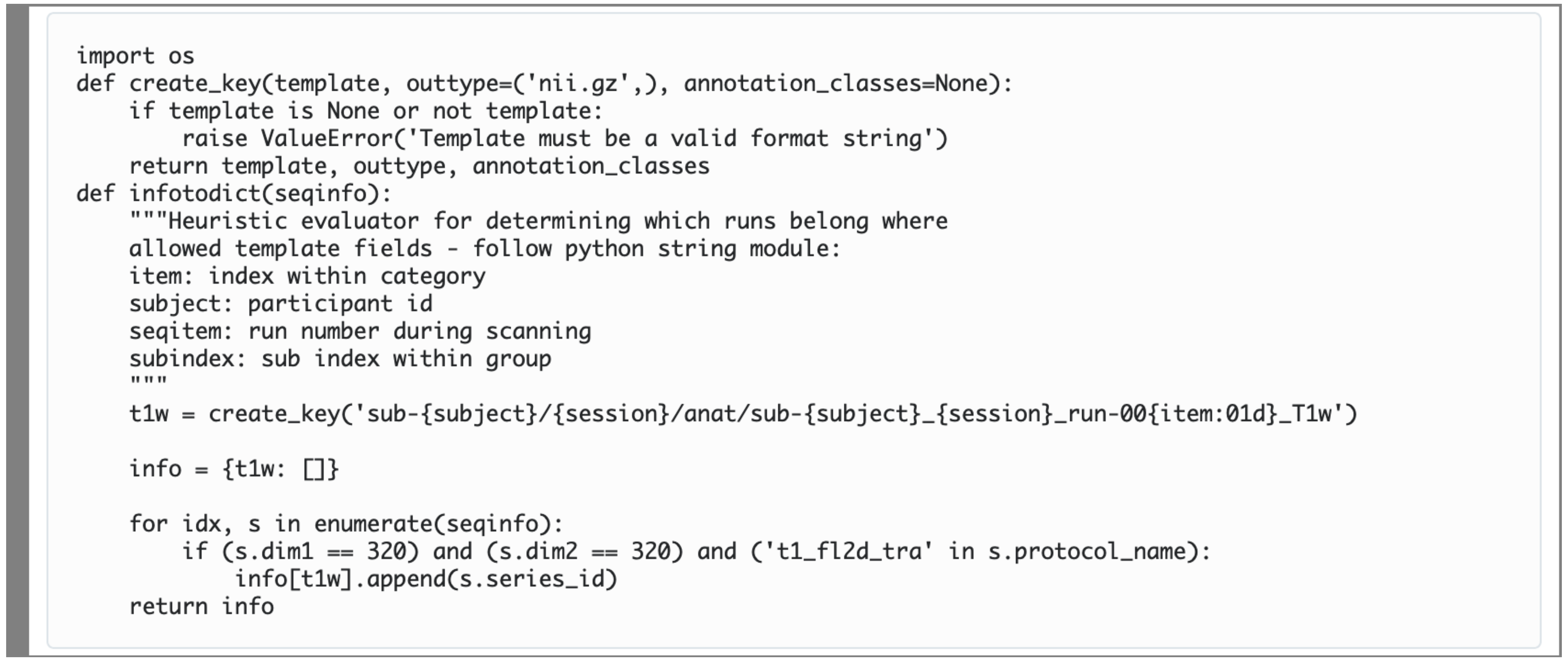
HeuDiConv

Recommended workflow

- Locally, install just the Python heudiconv package (don't need dcm2niix)
- Follow the walkthrough to get a scan spreadsheet (see results in hidden `.heudiconv` directory)
- Copy the template heuristic file and customize it for your scans
- Run heudiconv with your heuristic file with no conversion (`–c none`) to see how scans will be named
- Can use the Python debugger to better understand how the code runs (add `breakpoint()` in the heuristic file)
- On the cluster, use the heuristic file to run the actual conversion using Singularity (see Tesser and Bender projects for examples)



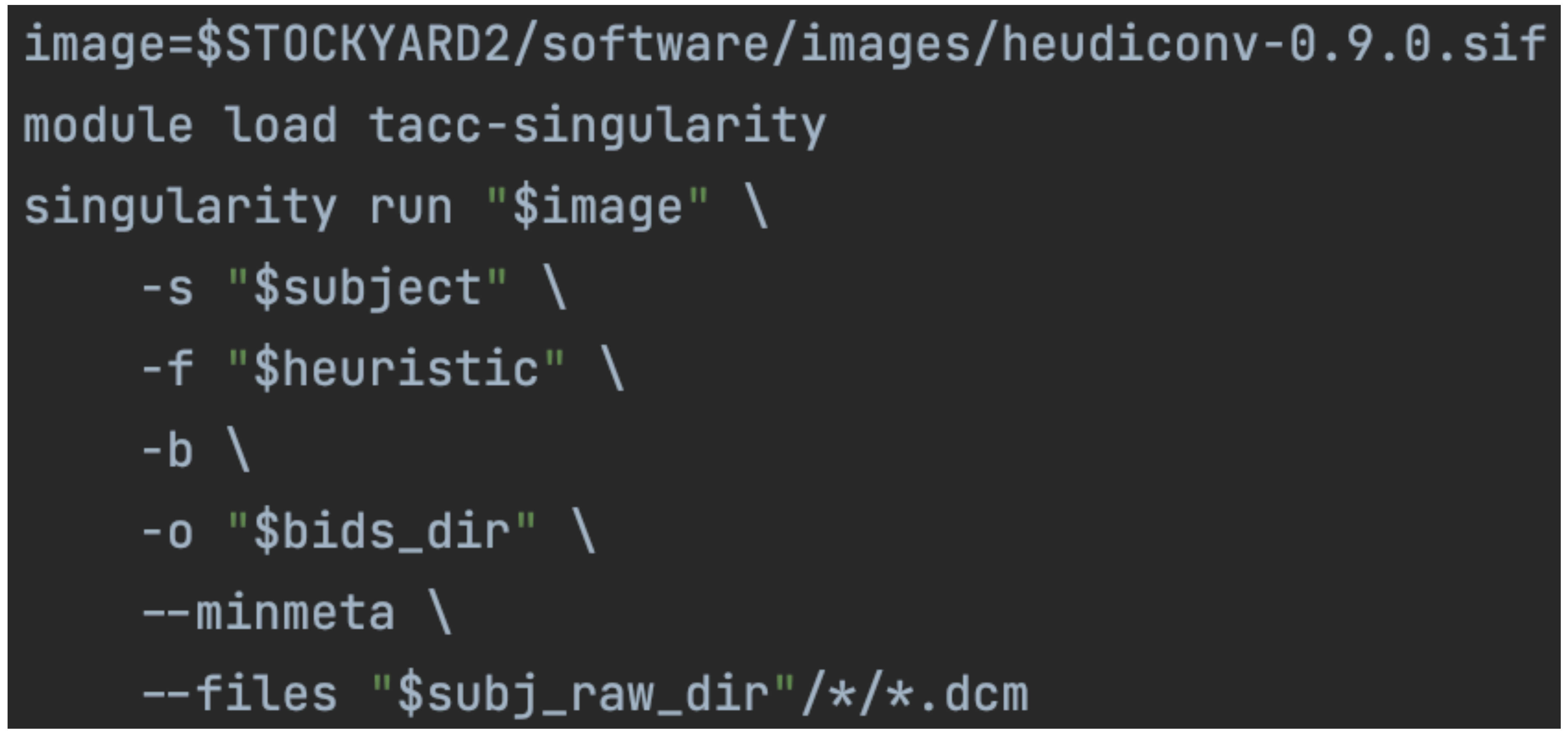
G	H	I	J	K	L	M
dim1	dim2	dim3	dim4	TR	TE	protocol_name
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra
320	320	10	1	0.15	3.74	t1_fl2d_tra



```
import os
def create_key(template, outtype=('nii.gz',), annotation_classes=None):
    if template is None or not template:
        raise ValueError('Template must be a valid format string')
    return template, outtype, annotation_classes
def infotodict(seqinfo):
    """Heuristic evaluator for determining which runs belong where
    allowed template fields - follow python string module:
    item: index within category
    subject: participant id
    seqitem: run number during scanning
    subindex: sub index within group
    """
    t1w = create_key('sub-{subject}/{session}/anat/sub-{subject}_{session}_run-00{item:01d}_T1w')

    info = {t1w: []}

    for idx, s in enumerate(seqinfo):
        if (s.dim1 == 320) and (s.dim2 == 320) and ('t1_fl2d_tra' in s.protocol_name):
            info[t1w].append(s.series_id)
    return info
```

Running HeuDiConv on TACC

- On TACC clusters like Lonestar 6, can only run Singularity on compute nodes, not login nodes
- Load the `tacc-singularity` module
- Specify the path to a container or "image", followed by the inputs to the program
- Containers have entry points that work the same as scripts (here, same as running the `heudiconv` command)



```
image=$STOCKYARD2/software/images/heudiconv-0.9.0.sif
module load tacc-singularity
singularity run "$image" \
    -s "$subject" \
    -f "$heuristic" \
    -b \
    -o "$bids_dir" \
    --minmeta \
    --files "$subj_raw_dir"/*/*.dcm
```

HeuDiConv tips

- Convert both the BOLD timeseries images and the "SBRef" (single-band reference scans)
  - fMRIprep can use SBRef scans for improved registration between functional and anatomical scans
- Exclude any scans you won't analyze (e.g., aborted scans) at this stage. Can check the number of volumes and exclude scans with less than the expected number
- Can program exceptions in the heuristic file (e.g., if `patient_id` is 102, exclude the first coronal scan)
- Can run into problems with multi-day studies, because scan order is determined based on a number assigned within the scanning session. Must first edit this number to avoid problems (have a Python script to do this in the Bender GitHub project)

After conversion

- After conversion, need to set the IntendedFor field for the JSON files associated with each fieldmap scan
- fMRIprep uses this information when correcting for distortion of functional scans
- The PyBIDS package can help work with BIDS datasets. The Tesser and Bender GitHub projects use PyBIDS to add IntendedFor information to the fieldmap scans

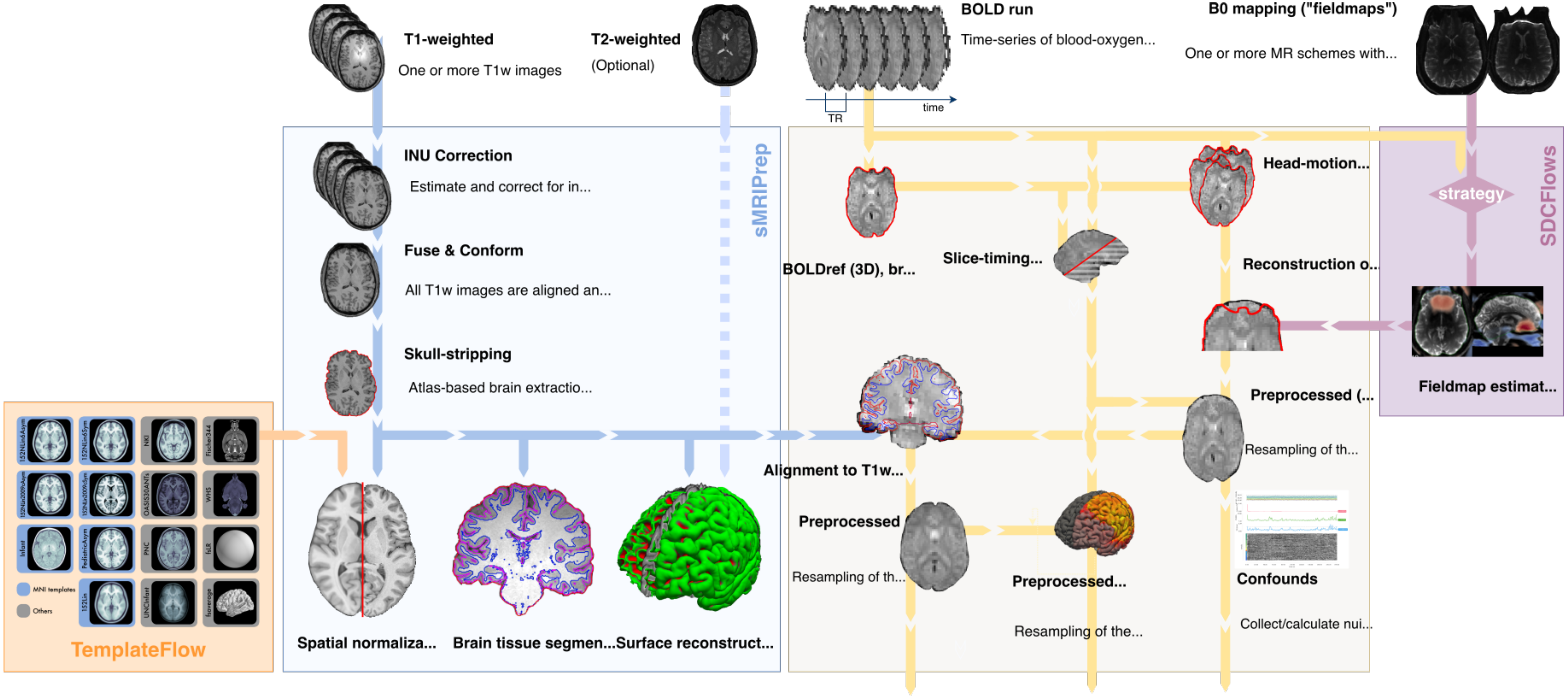
```
"IntendedFor": [
    "func/sub-30_task-localizer_run-1_bold.nii.gz",
    "func/sub-30_task-localizer_run-2_bold.nii.gz",
    "func/sub-30_task-localizer_run-3_bold.nii.gz",
    "func/sub-30_task-localizer_run-4_bold.nii.gz",
    "func/sub-30_task-exposure_run-1_bold.nii.gz"
]
```

Validating a dataset

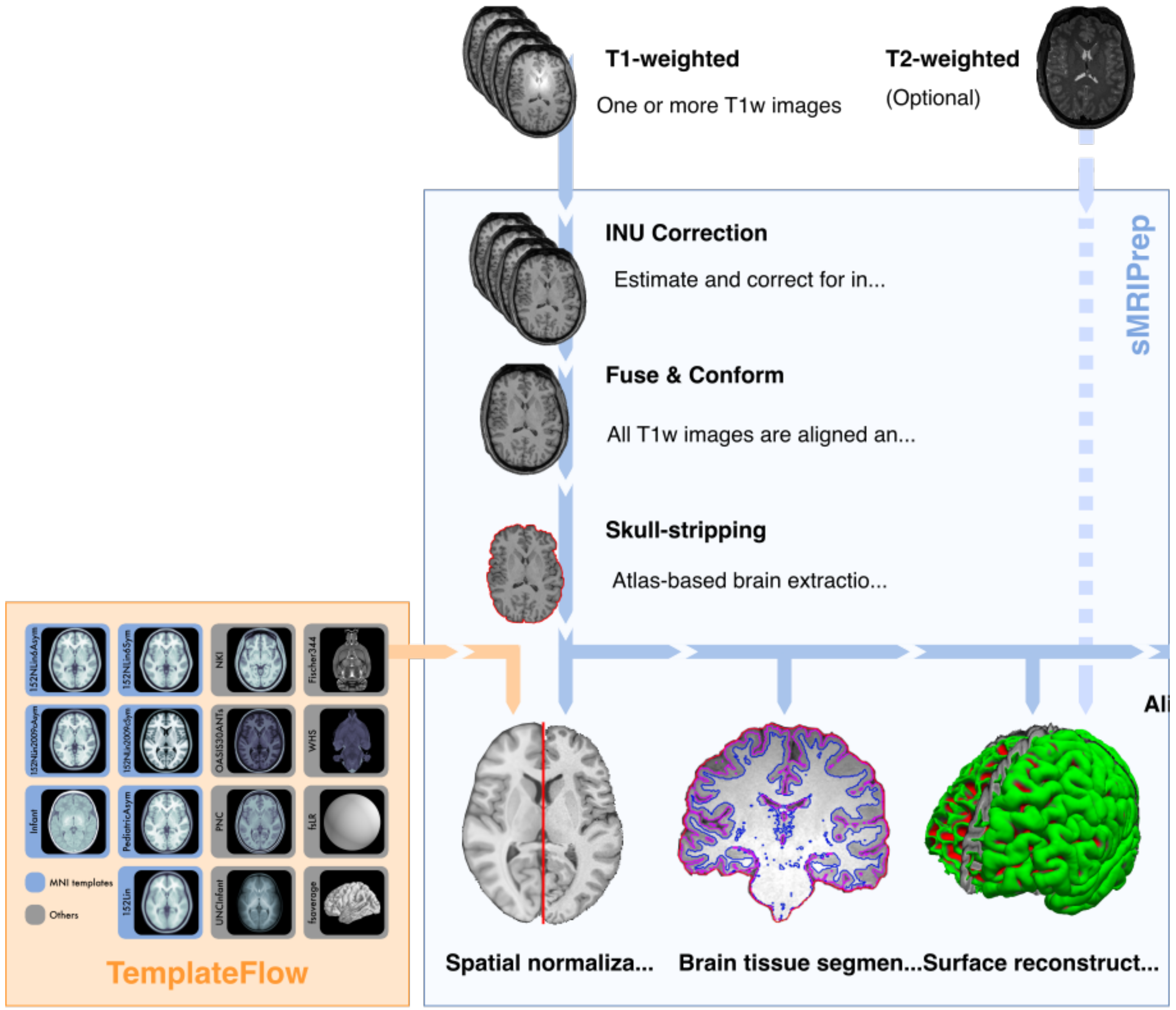
- Can use the BIDS-Validator program to check a converted dataset
  - `singularity run /work/03206/mortonne/software/images/validator-1.7.3.sif path_to_bids_directory`
- Will flag missing or mismatching files (e.g., if participants have different scans)
- May need to adjust your heuristic file and run HeuDiConv again

fMRIprep

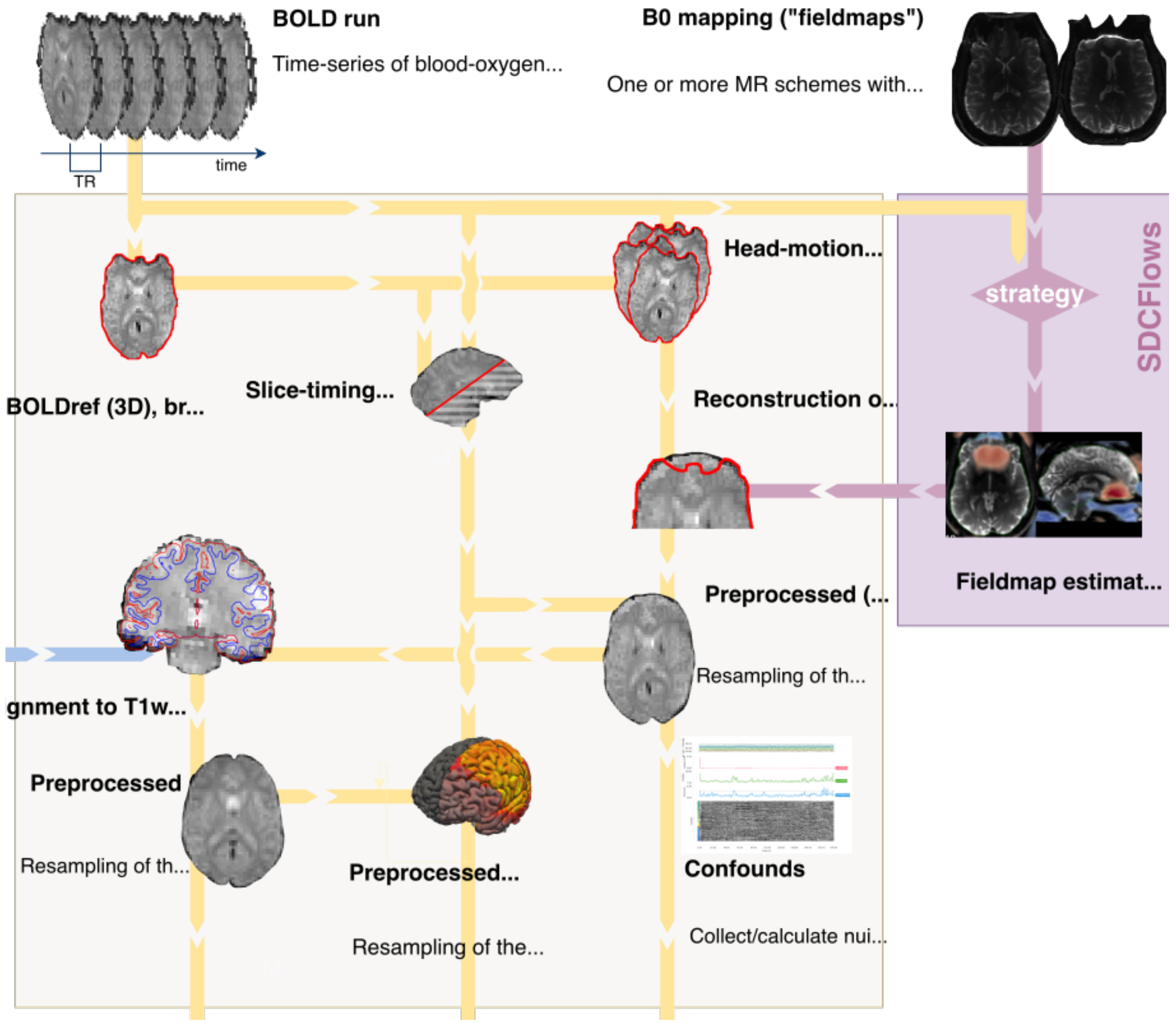
fMRIprep workflow



Preprocessing steps



Preprocessing steps



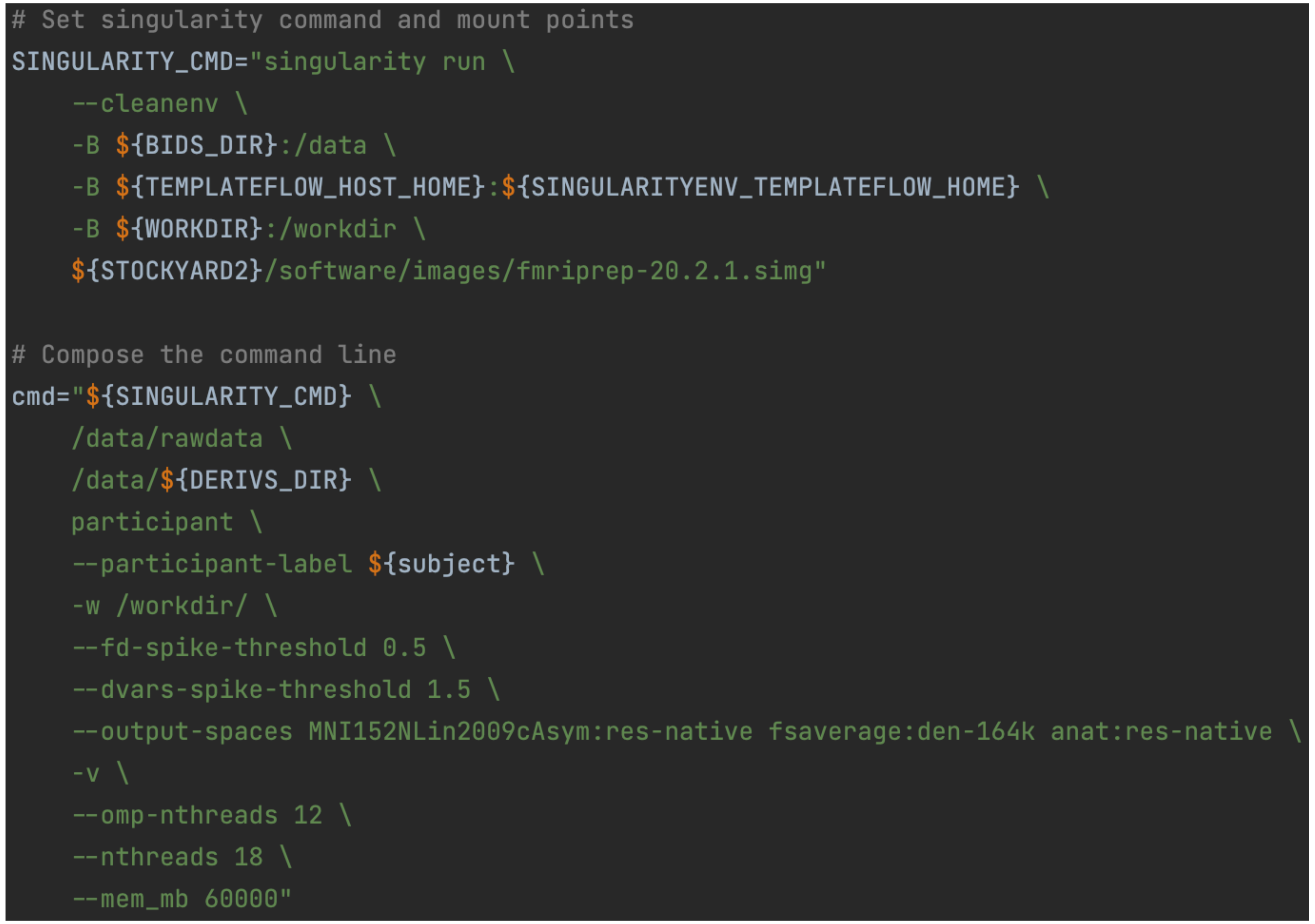
Organizing data

- Have three main directories:
  - sourcedata - raw DICOM files, raw behavioral files, etc.
  - rawdata - BIDS format neural and behavioral data
  - derivatives - results from analyzing BIDS data

```
├── derivatives
│   ├── fmriprep-20.2.7
│   ├── fmriprep-21.0.1
│   └── mriqc-0.16.1
├── rawdata
│   ├── CHANGES
│   ├── dataset_description.json
│   ├── participants.json
│   ├── participants.tsv
│   ├── README
│   ├── scans.json
│   ├── sub-01
│   ├── sub-02
│   ├── sub-03
│   ├── sub-04
│   ├── sub-05
│   ├── sub-06
│   ├── sub-07
│   ├── sub-08
│   ├── task-ABfeedback_events.json
│   ├── task-ABstudy_events.json
│   ├── task-ABtest_events.json
│   ├── task-ACtest_bold.json
│   ├── task-ACtest_events.json
│   ├── task-BCXYstudy_bold.json
│   ├── task-BCXYstudy_events.json
│   ├── task-BCXYtest_events.json
│   ├── task-exposure_bold.json
│   ├── task-exposure_events.json
│   ├── task-localizer_bold.json
│   └── task-localizer_events.json
└── sourcedata
    ├── behav
    ├── dicom
    └── stimuli.csv
```

Running fMRIprep

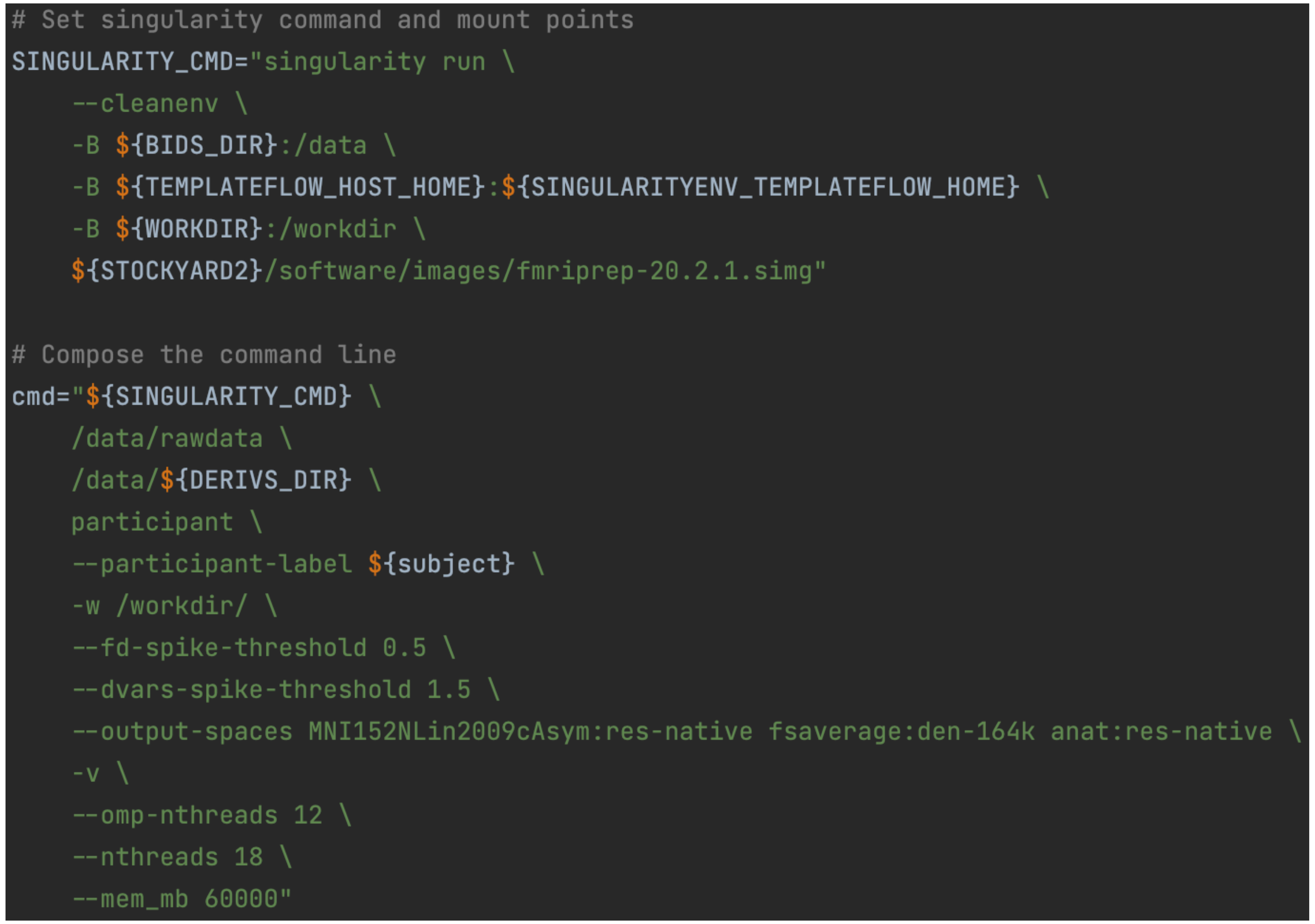
- The fMRIprep webpage has an example script to run it using Singularity
- Takes in BIDS format scan data
- Writes intermediate files to a "work" directory (not to be confused with the TACC `WORK` filesystem) when running
    - `$SCRATCH` a good filesystem to use for this
- In the derivatives directory, only outputs the data you request, without intermediate files
- Templates are downloaded to a TemplateFlow directory



```
# Set singularity command and mount points
SINGULARITY_CMD="singularity run \
    --cleanenv \
    -B ${BIDS_DIR}:/data \
    -B ${TEMPLATEFLOW_HOST_HOME}:${SINGULARITYENV_TEMPLATEFLOW_HOME} \
    -B ${WORKDIR}:/workdir \
    ${STOCKYARD2}/software/images/fmriprep-20.2.1.simg"

# Compose the command line
cmd="${SINGULARITY_CMD} \
    /data/rawdata \
    /data/${DERIVS_DIR} \
    participant \
    --participant-label ${subject} \
    -w /workdir/ \
    --fd-spike-threshold 0.5 \
    --dvars-spike-threshold 1.5 \
    --output-spaces MNI152NLin2009cAsym:res-native fsaverage:den-164k anat:res-native \
    -v \
    --omp-nthreads 12 \
    --nthreads 18 \
    --mem_mb 60000"
```

Output spaces

- Preprocessed data can be written in multiple output spaces
  - Native anatomical space
  - MNI template space
    - The MNI152NLin2009cAsym template is a better version of the FSL template
  - Surface space
    - fsaverage
    - fsLR (seems to be a newer version)
- Can choose the resolution (e.g., template space but in native resolution at 1.7 mm iso)



```
# Set singularity command and mount points
SINGULARITY_CMD="singularity run \
    --cleanenv \
    -B ${BIDS_DIR}:/data \
    -B ${TEMPLATEFLOW_HOST_HOME}:${SINGULARITYENV_TEMPLATEFLOW_HOME} \
    -B ${WORKDIR}:/workdir \
    ${STOCKYARD2}/software/images/fmriprep-20.2.1.simg"

# Compose the command line
cmd="${SINGULARITY_CMD} \
    /data/rawdata \
    /data/${DERIVS_DIR} \
    participant \
    --participant-label ${subject} \
    -w /workdir/ \
    --fd-spike-threshold 0.5 \
    --dvars-spike-threshold 1.5 \
    --output-spaces MNI152NLin2009cAsym:res-native fsaverage:den-164k anat:res-native \
    -v \
    --omp-nthreads 12 \
    --nthreads 18 \
    --mem_mb 60000"
```

fMRIprep output

Quality assurance

- Derivatives like fMRIprep are formatted similarly to BIDS raw data
- Get an HTML report for quality assurance for each participant
- Download the html files and figures directories, then view the html files in a browser

```
fmriprep
├── dataset_description.json
├── desc-aparcaseg_dseg.tsv
├── desc-aseg_dseg.tsv
├── logs
│   ├── CITATION.bib
│   ├── CITATION.html
│   ├── CITATION.md
│   └── CITATION.tex
├── sub-01
│   ├── anat
│   ├── figures
│   ├── func
│   └── log
├── sub-01.html
├── sub-20
│   ├── anat
│   ├── figures
│   ├── func
│   └── log
├── sub-20.html
├── sub-25
│   ├── anat
│   ├── figures
│   ├── func
│   └── log
├── sub-25.html
├── sub-30
│   ├── anat
│   ├── figures
│   ├── func
│   └── log
└── sub-30.html
```

fMRIprep output

Anatomical

- FreeSurfer ROIs are in the dseg files
  - See desc-aseg_dseg.tsv in the main directory for labels
- Spatial transformations specified in .txt and .h5 files
- Surface geometry is in the .gii files
- If no space is indicated, it is in native space
- In each space:
  - T1 scan
  - Brain mask
  - Basic segmentation (gray matter, white matter, CSF)

```
fmriprep/sub-01/anat
├── sub-01_desc-aparcaseg_dseg.nii.gz
├── sub-01_desc-aseg_dseg.nii.gz
├── sub-01_desc-brain_mask.json
├── sub-01_desc-brain_mask.nii.gz
├── sub-01_desc-preproc_T1w.json
├── sub-01_desc-preproc_T1w.nii.gz
├── sub-01_dseg.nii.gz
├── sub-01_from-fsnative_to-T1w_mode-image_xfm.txt
├── sub-01_from-MNI152NLin2009cAsym_to-T1w_mode-image_xfm.h5
├── sub-01_from-T1w_to-fsnative_mode-image_xfm.txt
├── sub-01_from-T1w_to-MNI152NLin2009cAsym_mode-image_xfm.h5
├── sub-01_hemi-L_inflated.surf.gii
├── sub-01_hemi-L_midthickness.surf.gii
├── sub-01_hemi-L_pial.surf.gii
├── sub-01_hemi-L_smoothwm.surf.gii
├── sub-01_hemi-R_inflated.surf.gii
├── sub-01_hemi-R_midthickness.surf.gii
├── sub-01_hemi-R_pial.surf.gii
├── sub-01_hemi-R_smoothwm.surf.gii
├── sub-01_label-CSF_probseg.nii.gz
├── sub-01_label-GM_probseg.nii.gz
├── sub-01_label-WM_probseg.nii.gz
├── sub-01_run-1_from-orig_to-T1w_mode-image_xfm.txt
├── sub-01_run-2_from-orig_to-T1w_mode-image_xfm.txt
├── sub-01_space-MNI152NLin2009cAsym_desc-brain_mask.json
├── sub-01_space-MNI152NLin2009cAsym_desc-brain_mask.nii.gz
├── sub-01_space-MNI152NLin2009cAsym_desc-preproc_T1w.json
├── sub-01_space-MNI152NLin2009cAsym_desc-preproc_T1w.nii.gz
├── sub-01_space-MNI152NLin2009cAsym_dseg.nii.gz
├── sub-01_space-MNI152NLin2009cAsym_label-CSF_probseg.nii.gz
├── sub-01_space-MNI152NLin2009cAsym_label-GM_probseg.nii.gz
└── sub-01_space-MNI152NLin2009cAsym_label-WM_probseg.nii.gz
```

fMRIprep output

Functional

- Preprocessed BOLD data are in the desc-preproc_bold files
- Here, written in three spaces:
  - Native anatomical (useful for ROI analysis)
  - MNI template space (useful for univariate analysis and searchlights)
  - Surface space (can be used for both individual and group analysis)
- Also have confound information (motion parameters, CSF signal etc.)

```
fmriprep/sub-01/func
├── sub-01_task-ACtest_run-1_desc-confounds_timeseries.json
├── sub-01_task-ACtest_run-1_desc-confounds_timeseries.tsv
├── sub-01_task-ACtest_run-1_from-scanner_to-T1w_mode-image_xfm.txt
├── sub-01_task-ACtest_run-1_from-T1w_to-scanner_mode-image_xfm.txt
├── sub-01_task-ACtest_run-1_space-fsaverage_hemi-L_bold.func.gii
├── sub-01_task-ACtest_run-1_space-fsaverage_hemi-L_bold.json
├── sub-01_task-ACtest_run-1_space-fsaverage_hemi-R_bold.func.gii
├── sub-01_task-ACtest_run-1_space-fsaverage_hemi-R_bold.json
├── sub-01_task-ACtest_run-1_space-MNI152NLin2009cAsym_boldref.nii.gz
├── sub-01_task-ACtest_run-1_space-MNI152NLin2009cAsym_desc-aparcaseg_dseg.nii.gz
├── sub-01_task-ACtest_run-1_space-MNI152NLin2009cAsym_desc-aseg_dseg.nii.gz
├── sub-01_task-ACtest_run-1_space-MNI152NLin2009cAsym_desc-brain_mask.json
├── sub-01_task-ACtest_run-1_space-MNI152NLin2009cAsym_desc-brain_mask.nii.gz
├── sub-01_task-ACtest_run-1_space-MNI152NLin2009cAsym_desc-preproc_bold.json
├── sub-01_task-ACtest_run-1_space-MNI152NLin2009cAsym_desc-preproc_bold.nii.gz
├── sub-01_task-ACtest_run-1_space-T1w_boldref.nii.gz
├── sub-01_task-ACtest_run-1_space-T1w_desc-aparcaseg_dseg.nii.gz
├── sub-01_task-ACtest_run-1_space-T1w_desc-aseg_dseg.nii.gz
├── sub-01_task-ACtest_run-1_space-T1w_desc-brain_mask.json
├── sub-01_task-ACtest_run-1_space-T1w_desc-brain_mask.nii.gz
├── sub-01_task-ACtest_run-1_space-T1w_desc-preproc_bold.json
├── sub-01_task-ACtest_run-1_space-T1w_desc-preproc_bold.nii.gz
```

Conclusions

- BIDS format makes neuroimaging data better documented and easier to work with
- BIDS apps can run relatively simply without the user having to specify many options; Docker/Singularity containers simplify installation
- fMRIprep is supported by an active community of developers who regularly fix bugs and add new features
- fMRIprep output is ready for a number of analysis approaches