Remi Janet
remi.janet@queensu.ca

fMRIprep: A Robust Preprocessing Pipeline for fMRI Data

https://fmriprep.org/

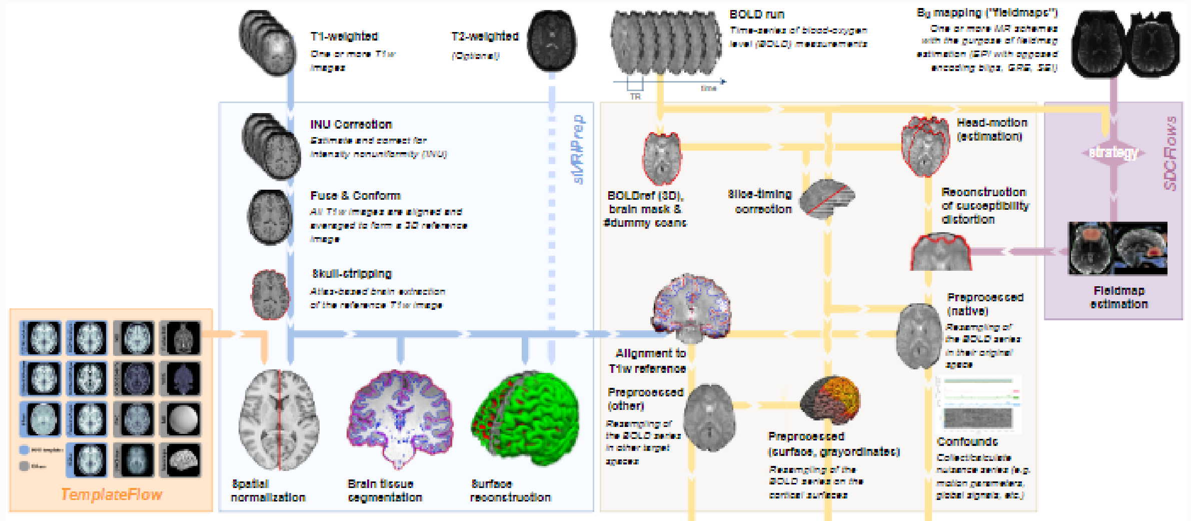
Remi Janet
remi.janet@queensu.ca

fMRIPrep: A Robust Preprocessing Pipeline for fMRI Data

fMRIPrep is a *NiPreps (NeuroImaging PREProcessing toolS)* application (www.nipreps.org) for the preprocessing of task-based and resting-state functional MRI (fMRI).

docker nipreps/fmriprep | Open in Code Ocean | circleci passing | docs passing | pypi v22.1.1 | doi 10.1038/s41592-018-0235-4 | RRID SCR_016216

About


T1-weighted
One or more T1w images
T2-weighted
(Optional)
BOLD run
Time-series of blood-oxygen level (BOLD) measurements
B0 mapping ("fieldmaps")
INU Correction
Estimate and correct for intensity nonuniformity (INU)
Fuse & Conform
All T1w images are aligned and averaged to form a 3D reference image
Skull-stripping
Atlas-based brain extraction of the reference T1w image
sMRIPrep
BOLDref (3D), brain mask & #dummy scans
Slice-timing correction
Head-motion (estimation)
Reconstruction of susceptibility distortion
strategy
SDCFlows
Fieldmap estimation
Alignment to T1w reference
Preprocessed (native)
Preprocessed (other)
Preprocessed (surface, grayordinates)
Confounds
TemplateFlow
Spatial normalization
Brain tissue segmentation
Surface reconstruction

- ▷ Installation
- ▷ BIDs format
- ▷ A brief overview of how to use fmriprep
  - Preprocessing steps – Running the analysis
  - Output – Examining the preprocessed data
  - Tips
- ▷ Conclusion/discussion

Installation

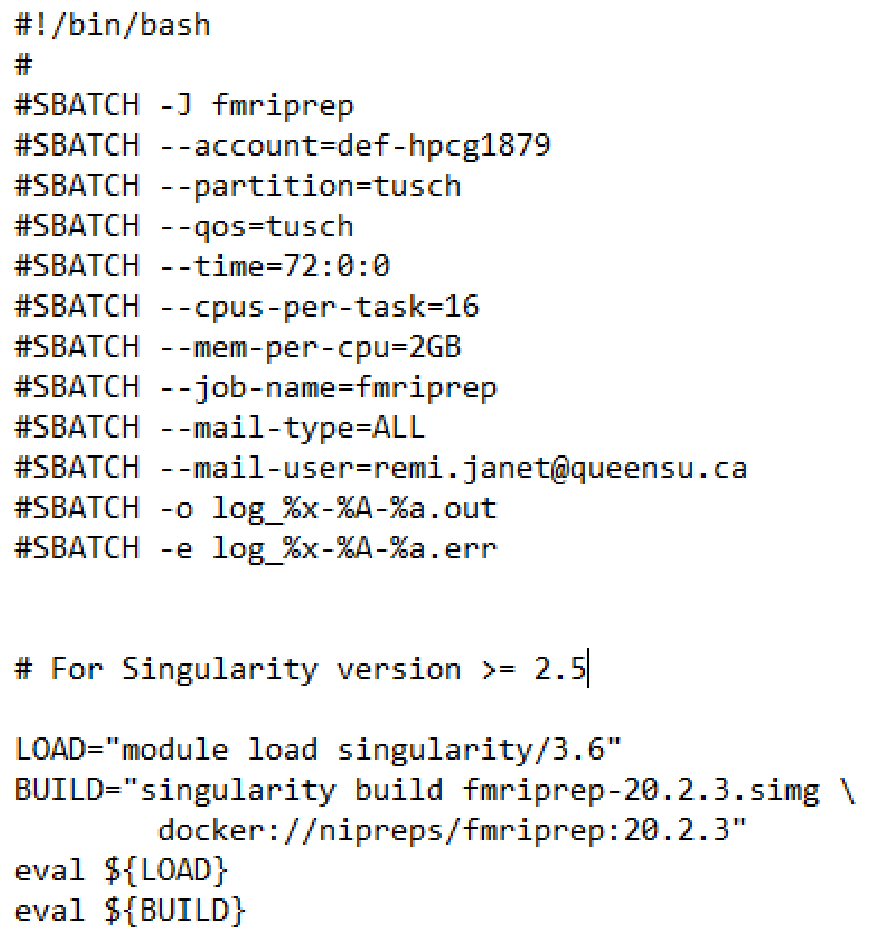
Installation

Installation

Singularity

Installation

Singularity



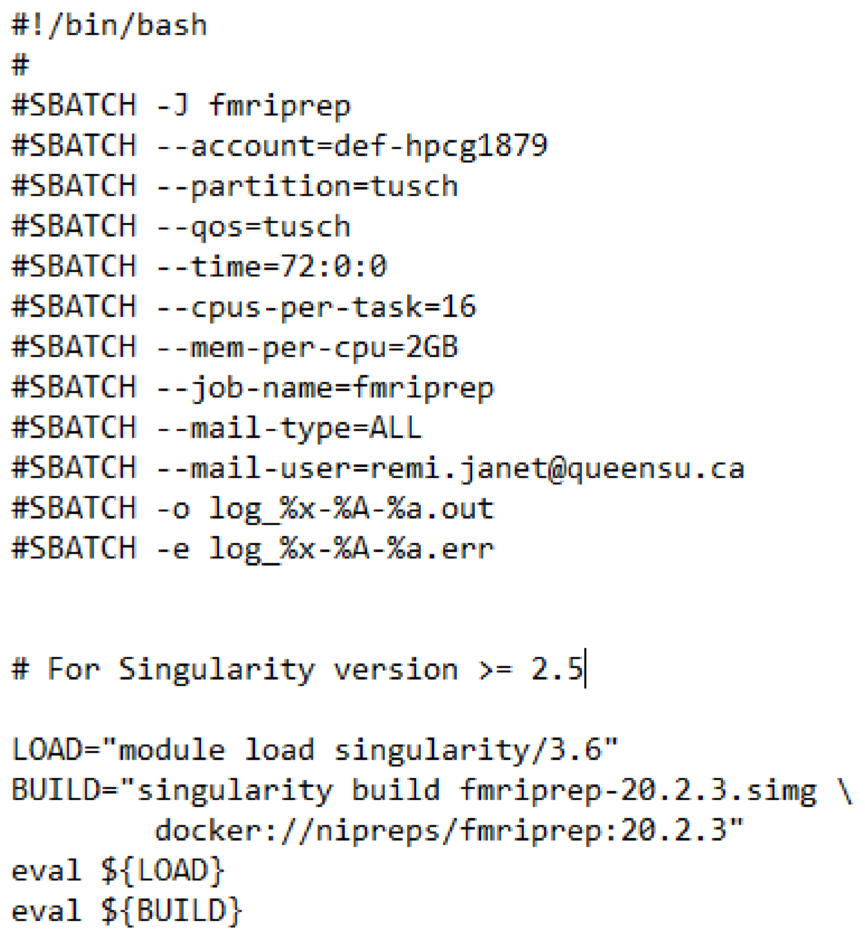
```
#!/bin/bash
#
#SBATCH -J fmriprep
#SBATCH --account=def-hpcg1879
#SBATCH --partition=tusch
#SBATCH --qos=tusch
#SBATCH --time=72:0:0
#SBATCH --cpus-per-task=16
#SBATCH --mem-per-cpu=2GB
#SBATCH --job-name=fmriprep
#SBATCH --mail-type=ALL
#SBATCH --mail-user=remi.janet@queensu.ca
#SBATCH -o log_%x-%A-%a.out
#SBATCH -e log_%x-%A-%a.err


# For Singularity version >= 2.5

LOAD="module load singularity/3.6"
BUILD="singularity build fmriprep-20.2.3.simg \
        docker://nipreps/fmriprep:20.2.3"
eval ${LOAD}
eval ${BUILD}
```

https://www.nipreps.org/apps/singularity/

Installation

Singularity

Python 3.7+

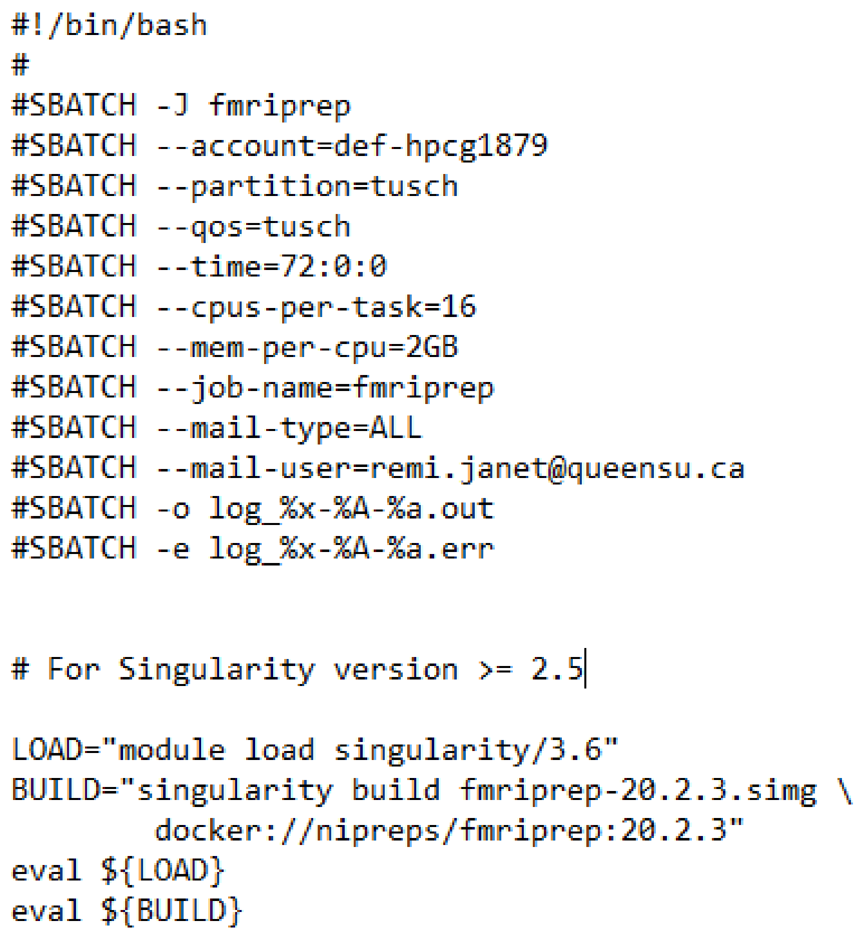


```
#!/bin/bash
#
#SBATCH -J fmriprep
#SBATCH --account=def-hpcg1879
#SBATCH --partition=tusch
#SBATCH --qos=tusch
#SBATCH --time=72:0:0
#SBATCH --cpus-per-task=16
#SBATCH --mem-per-cpu=2GB
#SBATCH --job-name=fmriprep
#SBATCH --mail-type=ALL
#SBATCH --mail-user=remi.janet@queensu.ca
#SBATCH -o log_%x-%A-%a.out
#SBATCH -e log_%x-%A-%a.err


# For Singularity version >= 2.5

LOAD="module load singularity/3.6"
BUILD="singularity build fmriprep-20.2.3.simg \
        docker://nipreps/fmriprep:20.2.3"
eval ${LOAD}
eval ${BUILD}
```

https://www.nipreps.org/apps/singularity/

Installation

Singularity

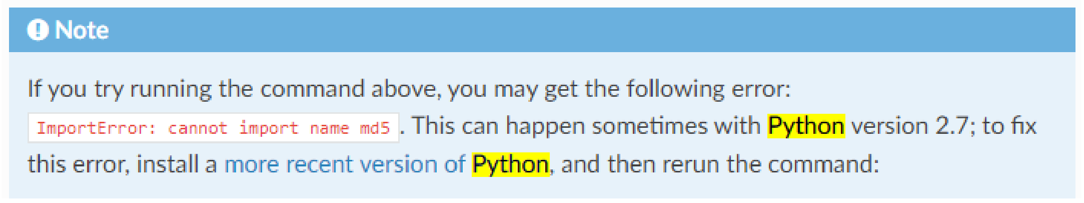


```
#!/bin/bash
#
#SBATCH -J fmriprep
#SBATCH --account=def-hpcg1879
#SBATCH --partition=tusch
#SBATCH --qos=tusch
#SBATCH --time=72:0:0
#SBATCH --cpus-per-task=16
#SBATCH --mem-per-cpu=2GB
#SBATCH --job-name=fmriprep
#SBATCH --mail-type=ALL
#SBATCH --mail-user=remi.janet@queensu.ca
#SBATCH -o log_%x-%A-%a.out
#SBATCH -e log_%x-%A-%a.err


# For Singularity version >= 2.5

LOAD="module load singularity/3.6"
BUILD="singularity build fmriprep-20.2.3.simg \
        docker://nipreps/fmriprep:20.2.3"
eval ${LOAD}
eval ${BUILD}
```

https://www.nipreps.org/apps/singularity/

Python 3.7+

```
$ python -m pip install fmriprep

# Check your installation
$ fmriprep --version
```



Note

If you try running the command above, you may get the following error: `ImportError: cannot import name md5`. This can happen sometimes with Python version 2.7; to fix this error, install a more recent version of Python, and then rerun the command:

https://fmriprep.org/en/stable/installation.html

Installation

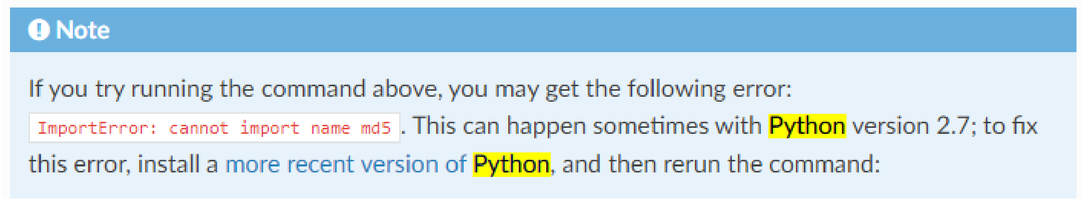
Singularity

```
#!/bin/bash
#
#SBATCH -J fmriprep
#SBATCH --account=def-hpcg1879
#SBATCH --partition=tusch
#SBATCH --qos=tusch
#SBATCH --time=72:0:0
#SBATCH --cpus-per-task=16
#SBATCH --mem-per-cpu=2GB
#SBATCH --job-name=fmriprep
#SBATCH --mail-type=ALL
#SBATCH --mail-user=remi.janet@queensu.ca
#SBATCH -o log_%x-%A-%a.out
#SBATCH -e log_%x-%A-%a.err


# For Singularity version >= 2.5

LOAD="module load singularity/3.6"
BUILD="singularity build fmriprep-20.2.3.simg \
        docker://nipreps/fmriprep:20.2.3"
eval ${LOAD}
eval ${BUILD}
```

https://www.nipreps.org/apps/singularity/

Python 3.7+

```
$ python -m pip install fmriprep

# Check your installation
$ fmriprep --version
```



Note

If you try running the command above, you may get the following error: `ImportError: cannot import name md5`. This can happen sometimes with Python version 2.7; to fix this error, install a more recent version of Python, and then rerun the command:

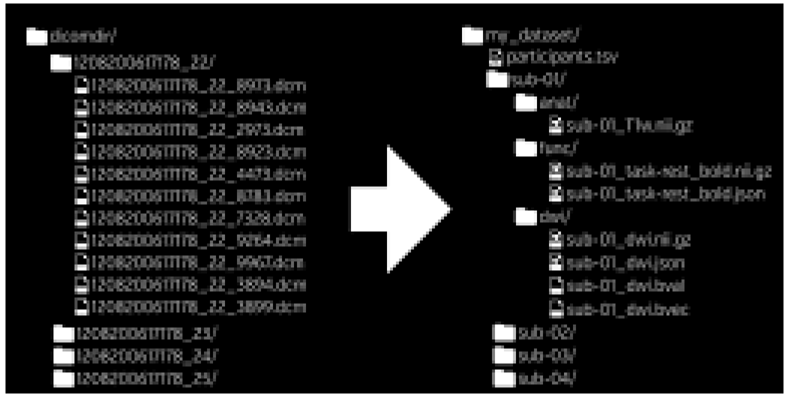
https://fmriprep.org/en/stable/installation.html

Docker

https://www.nipreps.org/apps/docker/

https://andysbrainbook.readthedocs.io/en/latest/OpenScience/OS/fMRIPrep.html#fmriprep

BIDs format



```
dicomdir/
    12082006171178_22/
        12082006171178_22_8973.dcm
        12082006171178_22_8943.dcm
        12082006171178_22_2973.dcm
        12082006171178_22_8923.dcm
        12082006171178_22_4473.dcm
        12082006171178_22_8783.dcm
        12082006171178_22_7328.dcm
        12082006171178_22_9264.dcm
        12082006171178_22_9967.dcm
        12082006171178_22_3894.dcm
        12082006171178_22_3899.dcm
    12082006171178_23/
    12082006171178_24/
    12082006171178_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/
```

https://andysbrainbook.readthedocs.io/en/latest/OpenScience/OS/BIDS_Overview.html

BIDs format

BIDS (Brain Imaging Data Structure)

Standarized format for the organization and description of neuroimaging and corresponding behavioral data.

Easily **shared** and **understood** by other researchers.
Reproducibility and Data Sharing

Different packages that can be used to convert your data into BIDS format, such as
dcm2bids, heudiconv, bidscoin, bidskit, etc.

BIDs format

Example with dcm2bids

1- Create Project folder
2- Download data
3- CMake and pip Installation
4- dcm2niix Installation
5- dcm2bids Installation
6- Setting Up Your Configuration File
7- Running the dcm2bids command
8- Double-check BIDS

https://unfmontreal.github.io/Dcm2Bids/3.0.2/#major-upgrade-with-dcm2bids-300

BIDs format

Example with dcm2bids

1- Create Project folder

```
cd $HOME
mkdir BIDS_tutorial
```

2- Download data

The data can be found here. https://drive.google.com/file/d/1Gx4GdWJEvT5O-2MYjDoyJIiirSpVFV4O/view

```
mv /global/project/hpcg1879/Remi/BIDS_tutorial/OpenScience $HOME/BIDS_tutorial
```

BIDs format

Example with dcm2bids

3- CMake and pip Installation

If you are working on a HPC (cluster) then they are already installed.

```
which cmake
which pip
```

Otherwise, follow these steps

```
cd
curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
python get-pip.py --user
export PATH= "/Users/$USER/Library/Python/2.7/bin/:$PATH"
cd ~/Downloads
tar -zxvf cmake-3.16.3-*-x86_64.tar.gz
export PATH="~/Downloads/cmake-3.16.3-Darwin-x86_64/CMake.app/Contents/bin/:$PATH"
pip install cmake
```

BIDs format

Example with dcm2bids

4- dcm2niix Installation

```
cd ~
git clone https://github.com/rordenlab/dcm2niix.git
cd dcm2niix
mkdir build
cd build
cmake ..
make
```

Add dcm2nix to your path

```
export PATH="$HOME/dcm2niix/build/bin/:$PATH"
```

5- dcm2bids Installation

```
cd $HOME/BIDS_tutorial
module load python
pip install --user dcm2bids
```

BIDs format

Example with dcm2bids

6- Setting Up Your Configuration File

See the file I shared with you.
It is a **.txt file**. However, you can just save it and **change the extension to .json.**

```
{
                    "descriptions": [
                    {
                                        "dataType": "anat",
                                        "modalityLabel": "T1w",
                                        "criteria": {
                                                            "SidecarFilename": "002*"
                                        }
                    },
                    ....
```

7- Running the dcm2bids command

```
dcm2bids -d $HOME/BIDS_tutorial/BIDS_tutorial_data -p 001 -c $HOME/BIDS_config.json -o $HOME/BIDS_tutorial --forceDcm2niix
```

⚠ This is true if you only have one session per subject.
If you have multiple session then copy this line

```
dcm2bids -d $HOME/BIDS_tutorial/BIDS_tutorial_data -p 001 –s 01 -c $HOME/BIDS_config.json -o $HOME/BIDS_tutorial --forceDcm2niix
```

BIDs format

Example with dcm2bids

8- Double-check BIDS

You can use the BIDS validator to ensure that your data are BIDS-compliant

https://bids-standard.github.io/bids-validator/

BIDs format

Files

```
sub-101
  anat
  fmap
  func
```

Anatomical scan

- sub-101_T1w.nii.gz
- sub-101_T1w.json

Functional scans

- sub-102_task-choose_run-01_events.tsv
- sub-102_task-choose_run-01_bold.nii.gz
- sub-102_task-choose_run-01_bold.nii
- sub-102_task-choose_run-01_bold.json

Fieldmaps

- sub-101_phasediff.nii.gz
- sub-101_phasediff.json
- sub-101_magnitude2.nii.gz
- sub-101_magnitude2.json
- sub-101_magnitude1.nii.gz
- sub-101_magnitude1.json

Metadata files

- participants.tsv
- dataset_description.json
- bids_filter.json
- .bidsignore

dataset_description.json

```
{
    "Authors": [
        "Remi Janet",
        "John-Dennis (Jack) Parsons",
        "Anita Tusche",
        "Hilke Plassman"
    ],
    "Acknowledgements":"HP supplied raw data to AT which was converted to BIDS by JP. and RJ",
    "Name": "fMRI Food Regulation Study",
    "BIDSVersion": "1.6.0"
}
```

participants.tsv

```
participant_id  scanner_id      age     female
sub-101 CC0058  27      0
sub-102 CC0003  31      1
sub-103 CC0007  18      1
sub-107 CC0025  25      0
sub-108 CC0012  28      0
sub-109 CC0033  32      0
sub-110 CC0054  32      0
sub-112 CC0027  20      1
sub-114 CC0049  39      0
sub-116 CC0029  30      0
sub-118 CC0005  34      1
sub-120 CC0024  28      0
sub-122 CC0053  31      1
sub-123 CC0021  25      0
sub-126 CC0041  29      0
sub-130 CC0008  26      0
```

.bidsignore

```
*.html
logs/
figures/
*_xfm.*
*.surf.gii
*_boldref.nii.gz
*_bold.func.gii
*_mixing.tsv
*_AROMAnoiseICs.csv
*_timeseries.tsv
```

BIDs format

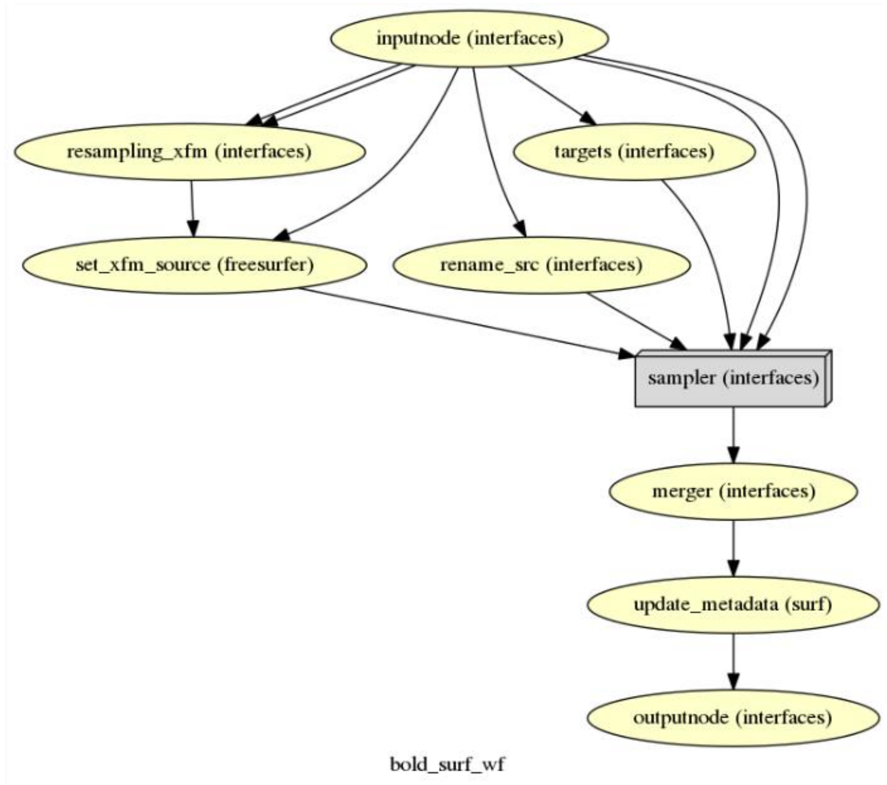
Useful links to go further:

- https://openneuro.org/
- https://www.mathworks.com/matlabcentral/fileexchange/42997-xiangruili-dicm2nii
- https://bids-apps.neuroimaging.io/
- https://bids-standard.github.io/bids-validator/
- https://openneuro.org/
- https://bids-standard.github.io/bids-starter-kit/dataset_examples.html

https://www.mathworks.com/matlabcentral/fileexchange/42997-xiangruili-dicm2nii

```
setpref('dicm2nii_gui_para', 'bidsForceGUI', true)
dicm2nii(DCfilesList, 'F:\path_to_your_files\rawfiles', 'bids')
```

fMRIprep usage



A brief overview of how to use fmriprep

Preprocessing steps – Running the analysis

A brief overview of how to use fmriprep

With slurm. Use the ".slurm" files and launch them with the command

```
sbatch --array=[sub_start-sub_end] fmriprep_scrip.slurm
```

A brief overview of how to use fmriprep

With slurm. Use the “.slurm” files and launch them with the command

sbatch --array=[sub_start-sub_end] fmriprep_scrip.slurm

```
export STUDY="/global/project/hpcg1879/Remi/Altruism_AT"
BIDS_DIR="$STUDY"
DERIVS_DIR="derivatives/fmriprep-20.2.1"
FS_DIR="${BIDS_DIR}/${DERIVS_DIR}/freesurfer"

# 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 ${DERIVS_DIR}

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

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

SINGULARITY_CMD="singularity run --cleanenv -B ${BIDS_DIR}:/Altruism_AT -B ${TEMPLATEFLOW_HOST_HOME}:${SINGULARITYENV_TEMPLATEFLOW_HOME} fmriprep-20.2.1.simg"

# 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" ${STUDY}/participants.tsv )

echo $subject
# Remove IsRunning files from FreeSurfer
# find ${FS_DIR}/sub-$subject/ -name "*IsRunning*" -delete

#
SINGULARITY_LOAD="module load singularity/3.8"

# DON"T FUCK WITH THIS LINE RIGHT NOW

RUN_FMRIPREP_CMD="${SINGULARITY_CMD} /Altruism_AT /Altruism_AT/${DERIVS_DIR} participant --participant-label ${subject} --skip_bids_validation --ignore fieldmaps --output-spaces MNI152NLin2009cAsym"
eval ${SINGULARITY_LOAD}
eval ${RUN_FMRIPREP_CMD}

# 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
```

A brief overview of how to use fmriprep

With slurm. Use the “.slurm” files and launch them with the command

sbatch --array=[sub_start-sub_end] fmriprep_scrip.slurm

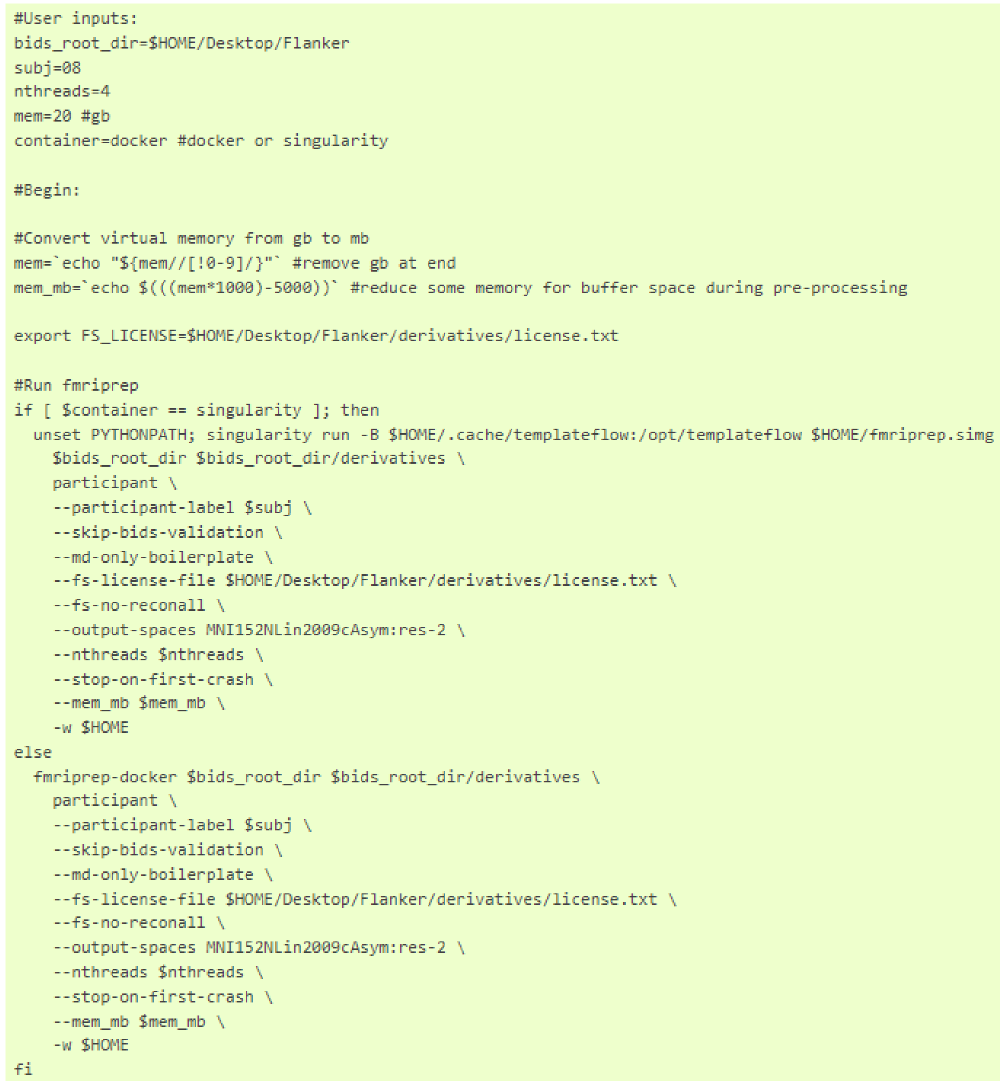
```
export STUDY="/global/project/hpcg1879/Remi/Altruism_AT"
BIDS_DIR="$STUDY"
DERIVS_DIR="derivatives/fmriprep-20.2.1"
FS_DIR="${BIDS_DIR}/${DERIVS_DIR}/freesurfer"

# 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 ${DERIVS_DIR}

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

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

SINGULARITY_CMD="singularity run --cleanenv -B ${BIDS_DIR}:/Altruism_AT -B ${TEMPLATEFLOW_HOST_HOME}:${SINGULARITYENV_TEMPLATEFLOW_HOME} fmriprep-20.2.1.simg"

# 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" ${STUDY}/participants.tsv )

echo $subject
# Remove IsRunning files from FreeSurfer
# find ${FS_DIR}/sub-$subject/ -name "*IsRunning*" -delete

#
SINGULARITY_LOAD="module load singularity/3.8"

# DON"T FUCK WITH THIS LINE RIGHT NOW

RUN_FMRIPREP_CMD="${SINGULARITY_CMD} /Altruism_AT /Altruism_AT/${DERIVS_DIR} participant --participant-label ${subject} --skip_bids_validation --ignore fieldmaps --output-spaces MNI152NLin2009cAsym"
eval ${SINGULARITY_LOAD}
eval ${RUN_FMRIPREP_CMD}

# 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
```

A brief overview of how to use fmriprep

With Python. Use bash code: bash frmiprepcode.sh

A brief overview of how to use fmriprep

With Python. Use bash code: bash frmiprepcode.sh

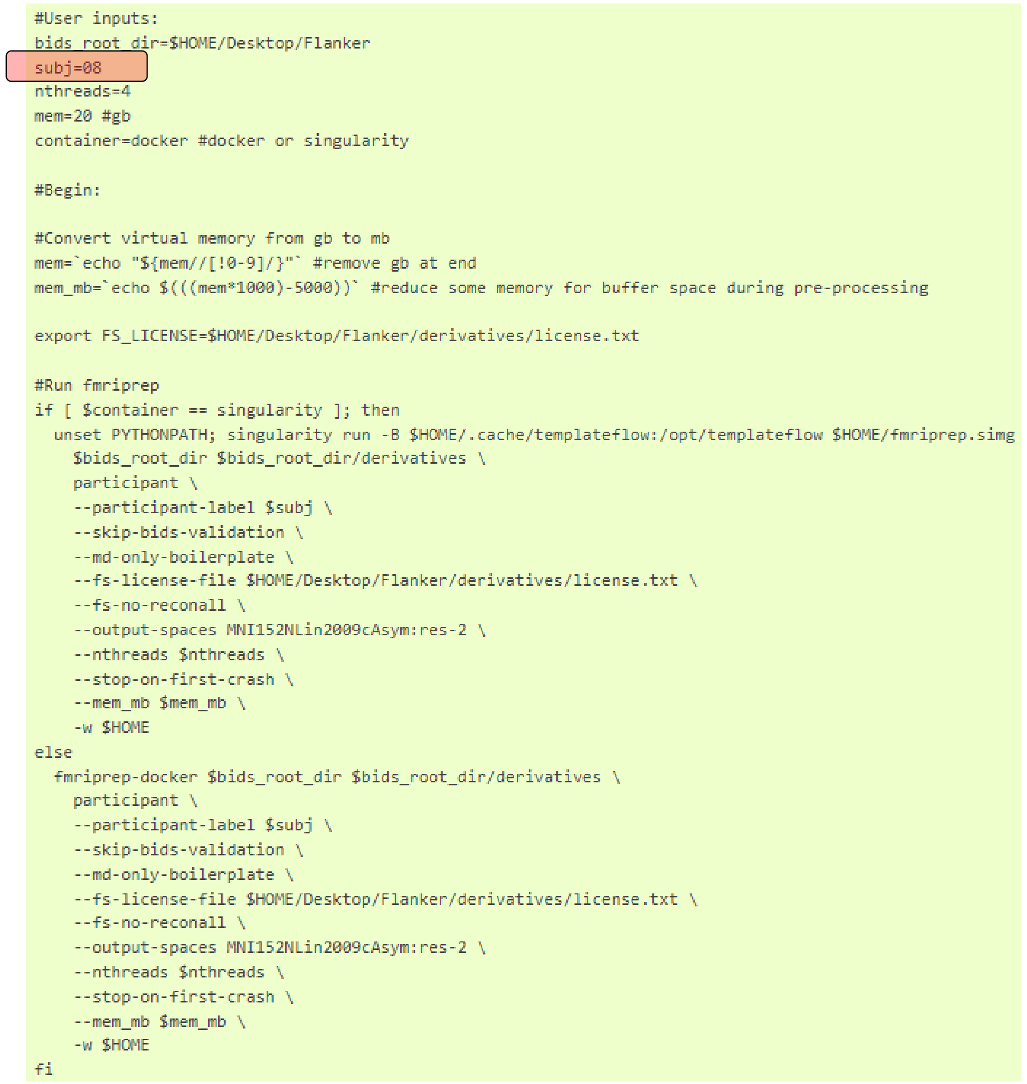


```
#User inputs:
bids_root_dir=$HOME/Desktop/Flanker
subj=08
nthreads=4
mem=20 #gb
container=docker #docker or singularity

#Begin:

#Convert virtual memory from gb to mb
mem=`echo "${mem//[!0-9]/}"` #remove gb at end
mem_mb=`echo $(((mem*1000)-5000))` #reduce some memory for buffer space during pre-processing

export FS_LICENSE=$HOME/Desktop/Flanker/derivatives/license.txt

#Run fmriprep
if [ $container == singularity ]; then
  unset PYTHONPATH; singularity run -B $HOME/.cache/templateflow:/opt/templateflow $HOME/fmriprep.simg
    $bids_root_dir $bids_root_dir/derivatives \
    participant \
    --participant-label $subj \
    --skip-bids-validation \
    --md-only-boilerplate \
    --fs-license-file $HOME/Desktop/Flanker/derivatives/license.txt \
    --fs-no-reconall \
    --output-spaces MNI152NLin2009cAsym:res-2 \
    --nthreads $nthreads \
    --stop-on-first-crash \
    --mem_mb $mem_mb \
    -w $HOME
else
  fmriprep-docker $bids_root_dir $bids_root_dir/derivatives \
    participant \
    --participant-label $subj \
    --skip-bids-validation \
    --md-only-boilerplate \
    --fs-license-file $HOME/Desktop/Flanker/derivatives/license.txt \
    --fs-no-reconall \
    --output-spaces MNI152NLin2009cAsym:res-2 \
    --nthreads $nthreads \
    --stop-on-first-crash \
    --mem_mb $mem_mb \
    -w $HOME
fi
```

https://andysbrainbook.readthedocs.io/en/latest/OpenScience/OS/fMRIPrep_Demo_2_RunningAnalysis.html

A brief overview of how to use fmriprep

With Python. Use bash code: bash frmiprepcode.sh



```
#User inputs:
bids_root_dir=$HOME/Desktop/Flanker
subj=08
nthreads=4
mem=20 #gb
container=docker #docker or singularity

#Begin:

#Convert virtual memory from gb to mb
mem=`echo "${mem//[!0-9]/}"` #remove gb at end
mem_mb=`echo $(((mem*1000)-5000))` #reduce some memory for buffer space during pre-processing

export FS_LICENSE=$HOME/Desktop/Flanker/derivatives/license.txt

#Run fmriprep
if [ $container == singularity ]; then
  unset PYTHONPATH; singularity run -B $HOME/.cache/templateflow:/opt/templateflow $HOME/fmriprep.simg
    $bids_root_dir $bids_root_dir/derivatives \
    participant \
    --participant-label $subj \
    --skip-bids-validation \
    --md-only-boilerplate \
    --fs-license-file $HOME/Desktop/Flanker/derivatives/license.txt \
    --fs-no-reconall \
    --output-spaces MNI152NLin2009cAsym:res-2 \
    --nthreads $nthreads \
    --stop-on-first-crash \
    --mem_mb $mem_mb \
    -w $HOME
else
  fmriprep-docker $bids_root_dir $bids_root_dir/derivatives \
    participant \
    --participant-label $subj \
    --skip-bids-validation \
    --md-only-boilerplate \
    --fs-license-file $HOME/Desktop/Flanker/derivatives/license.txt \
    --fs-no-reconall \
    --output-spaces MNI152NLin2009cAsym:res-2 \
    --nthreads $nthreads \
    --stop-on-first-crash \
    --mem_mb $mem_mb \
    -w $HOME
fi
```

It will process only subject 8 in this example.

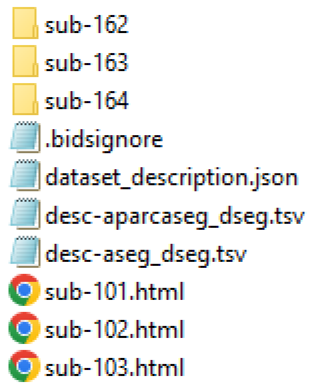
https://andysbrainbook.readthedocs.io/en/latest/OpenScience/OS/fMRIPrep_Demo_2_RunningAnalysis.html

A brief overview of how to use fmriprep

Output – Examining the preprocessed data

A brief overview of how to use fmriprep

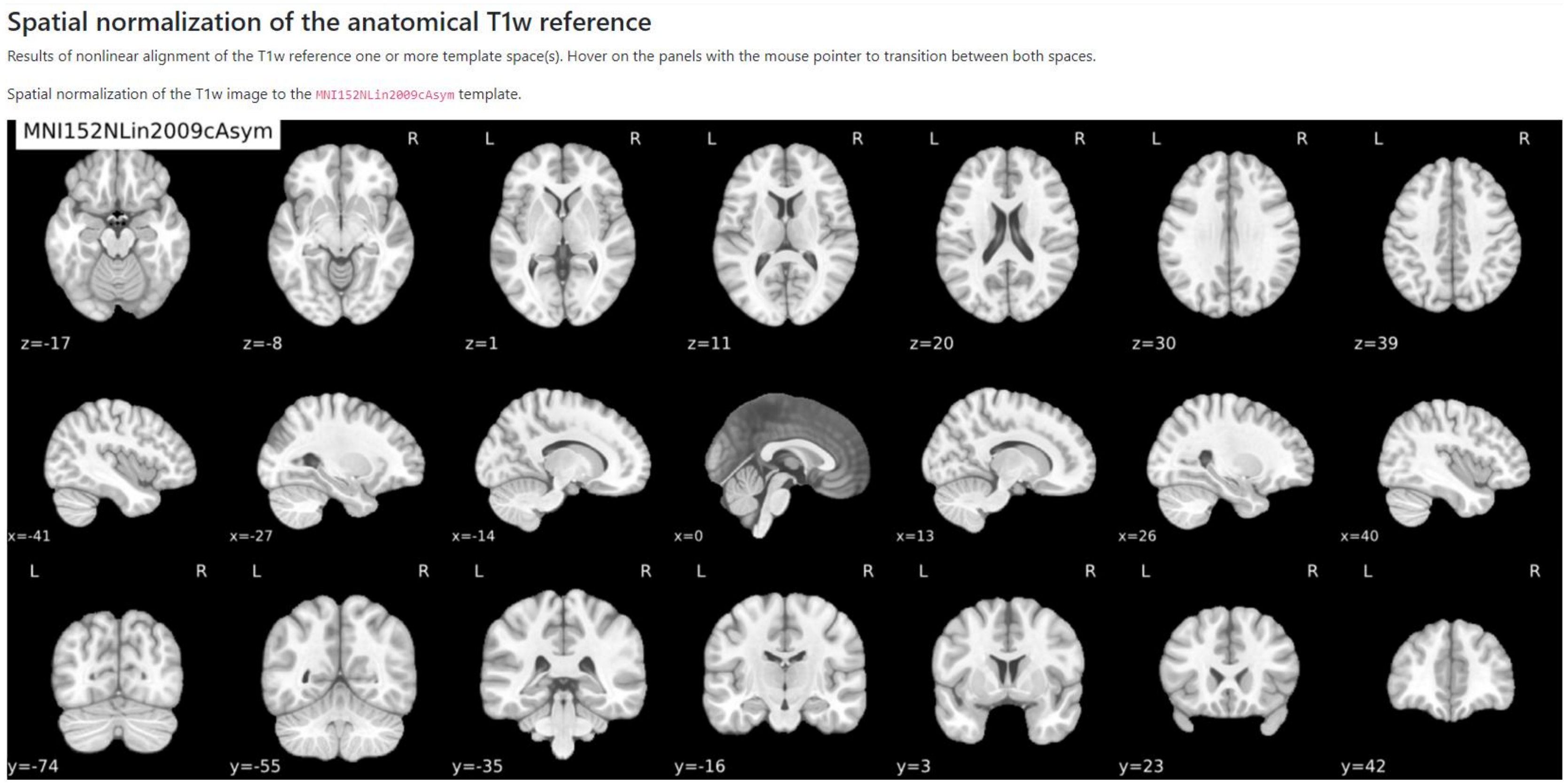
Output – Examining the preprocessed data



sub-162
sub-163
sub-164
.bidsignore
dataset_description.json
desc-aparcaseg_dseg.tsv
desc-aseg_dseg.tsv
sub-101.html
sub-102.html
sub-103.html

A brief overview of how to use fmriprep

Anatomical output



Spatial normalization of the anatomical T1w reference

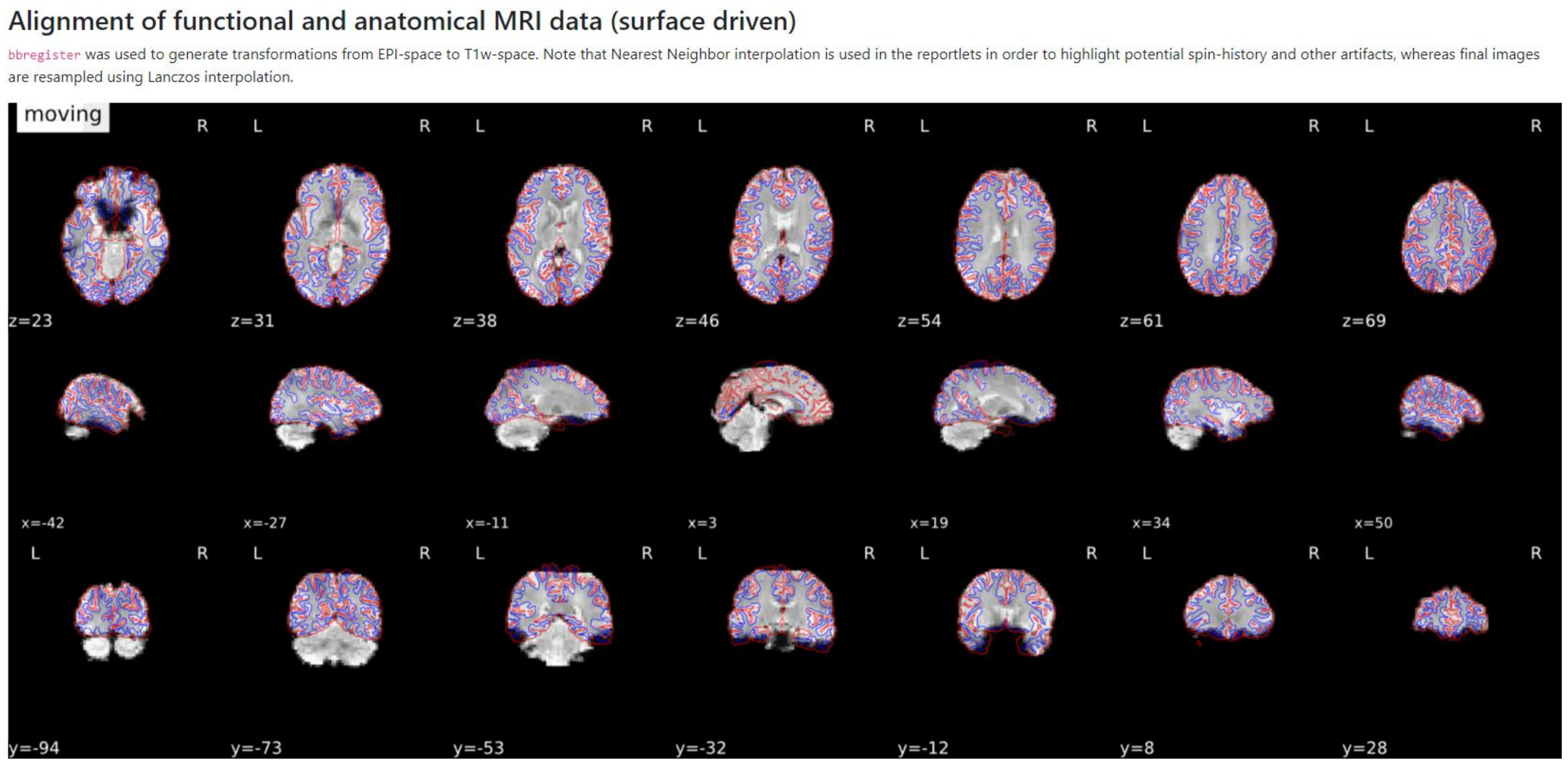
Results of nonlinear alignment of the T1w reference one or more template space(s). Hover on the panels with the mouse pointer to transition between both spaces.

Spatial normalization of the T1w image to the `MNI152NLin2009cAsym` template.

Make sure to check the alignment not only between the outlines of the brain, but also the internal structures such as the ventricles.

A brief overview of how to use fmriprep

Functional outputs



Alignment of functional and anatomical MRI data (surface driven)

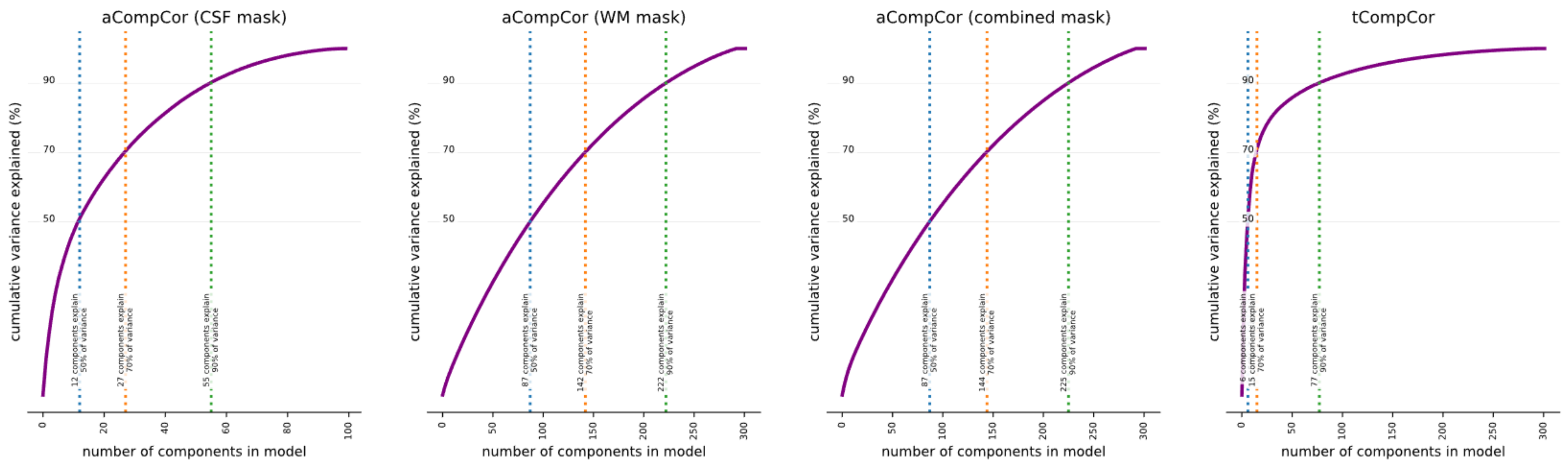
`bbregister` was used to generate transformations from EPI-space to T1w-space. Note that Nearest Neighbor interpolation is used in the reportlets in order to highlight potential spin-history and other artifacts, whereas final images are resampled using Lanczos interpolation.

Make sure that the internal structures are well aligned

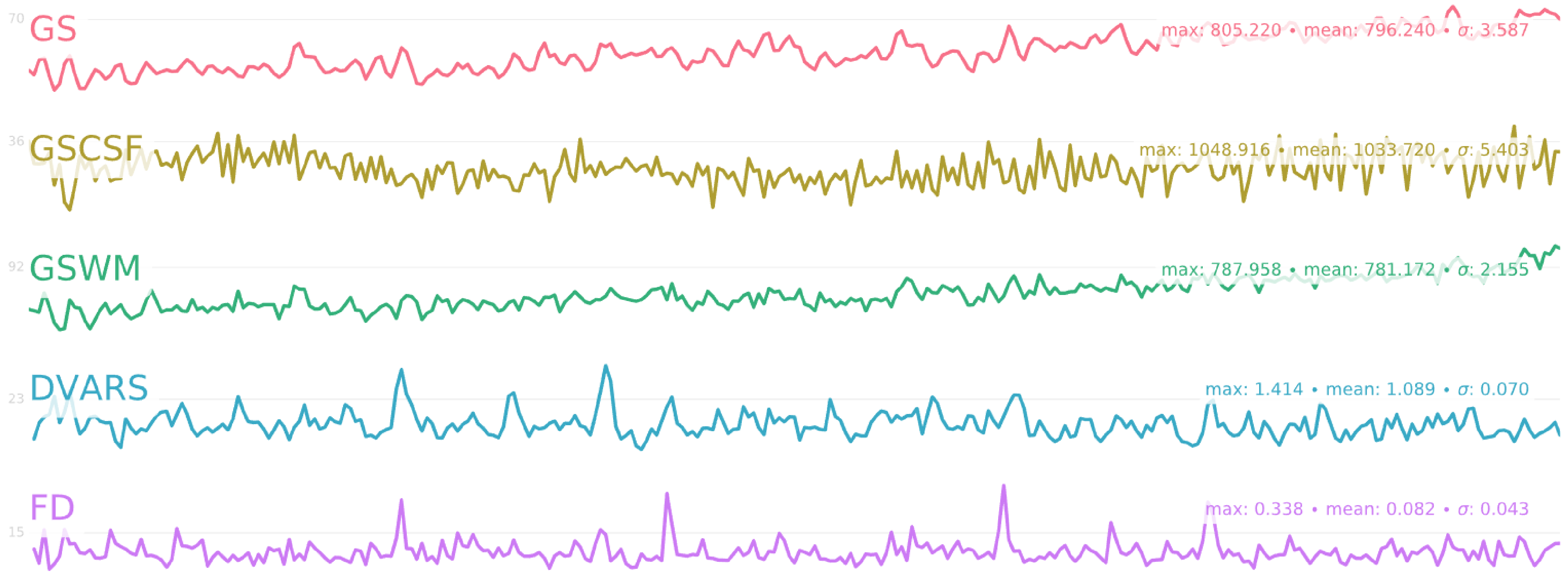
A brief overview of how to use fmriprep

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.

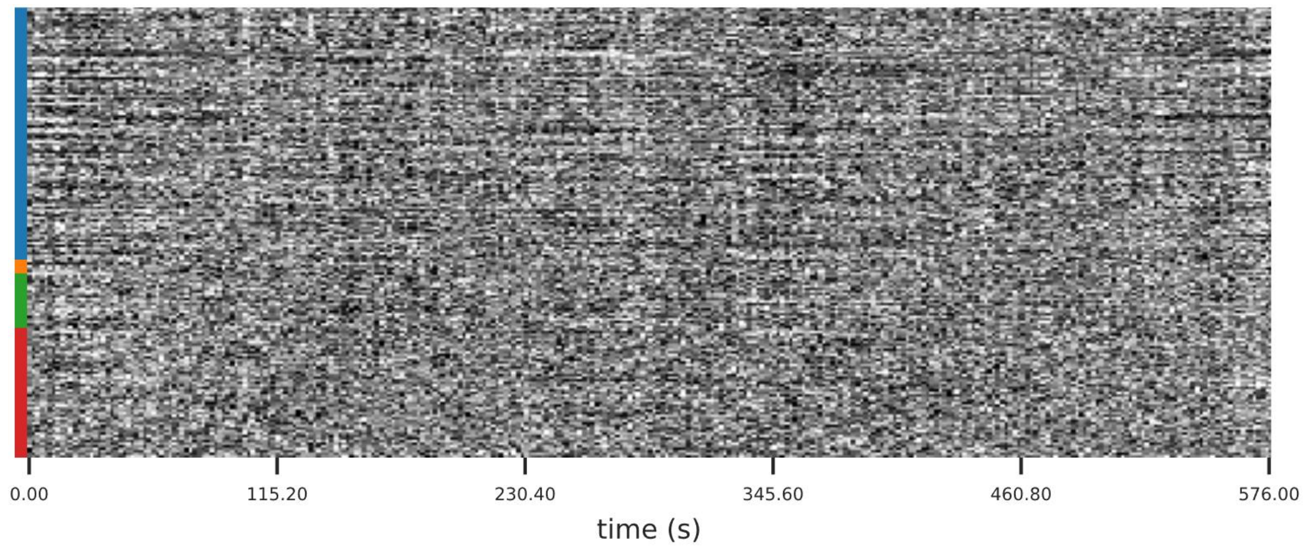


A brief overview of how to use fmriprep



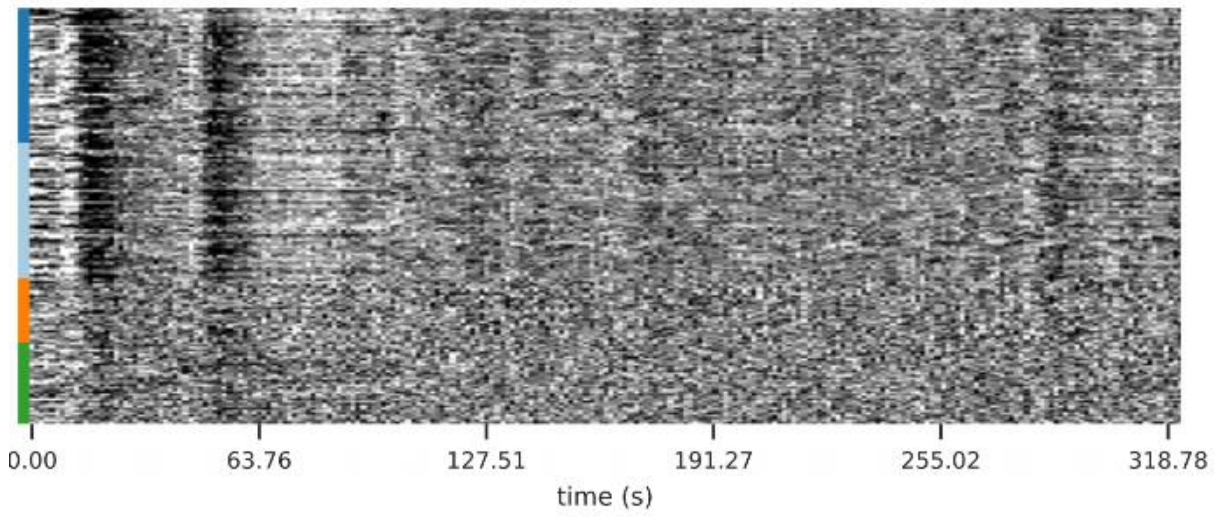
Rule of thumb = warning if mean FD (σFD) > 0.2

A brief overview of how to use fmriprep



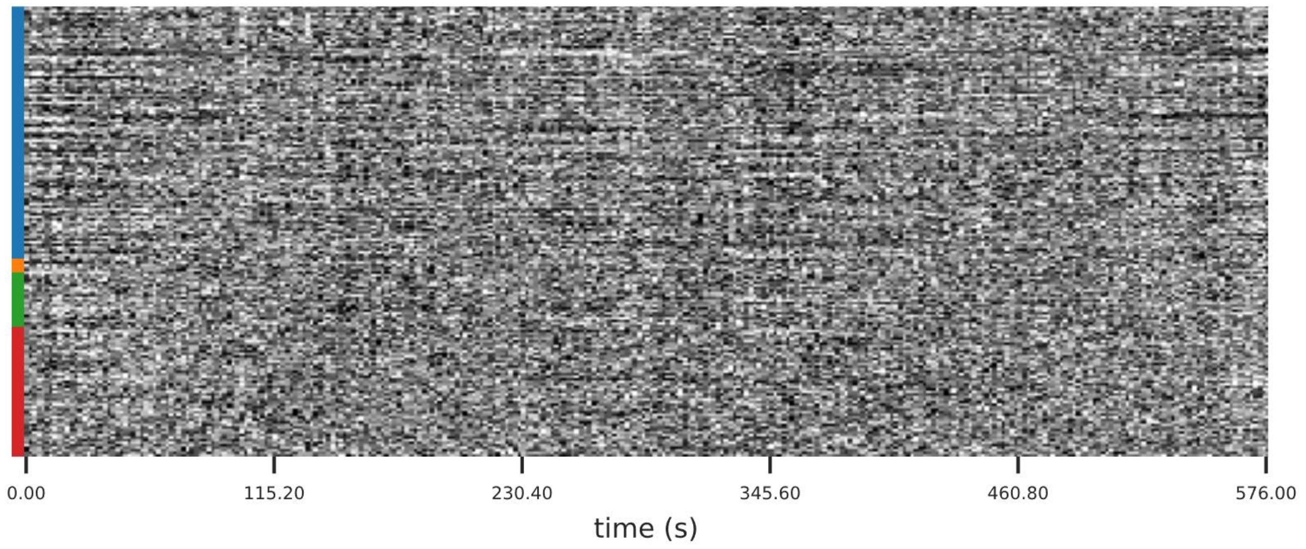
✓

Any sudden changes in motion may be reflected in uniform changes across the entire column for that timepoint.

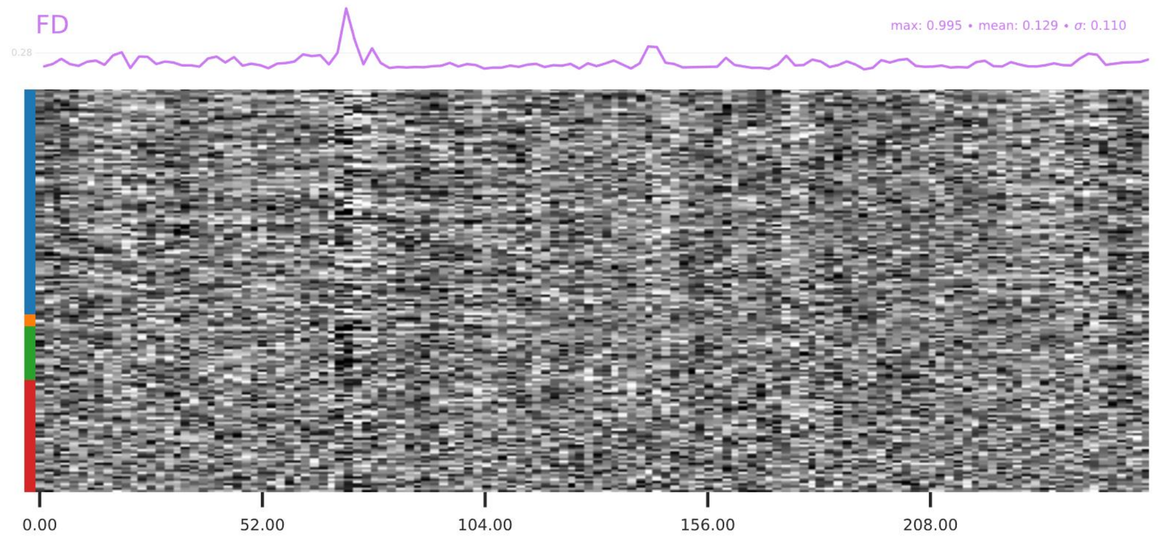


✗

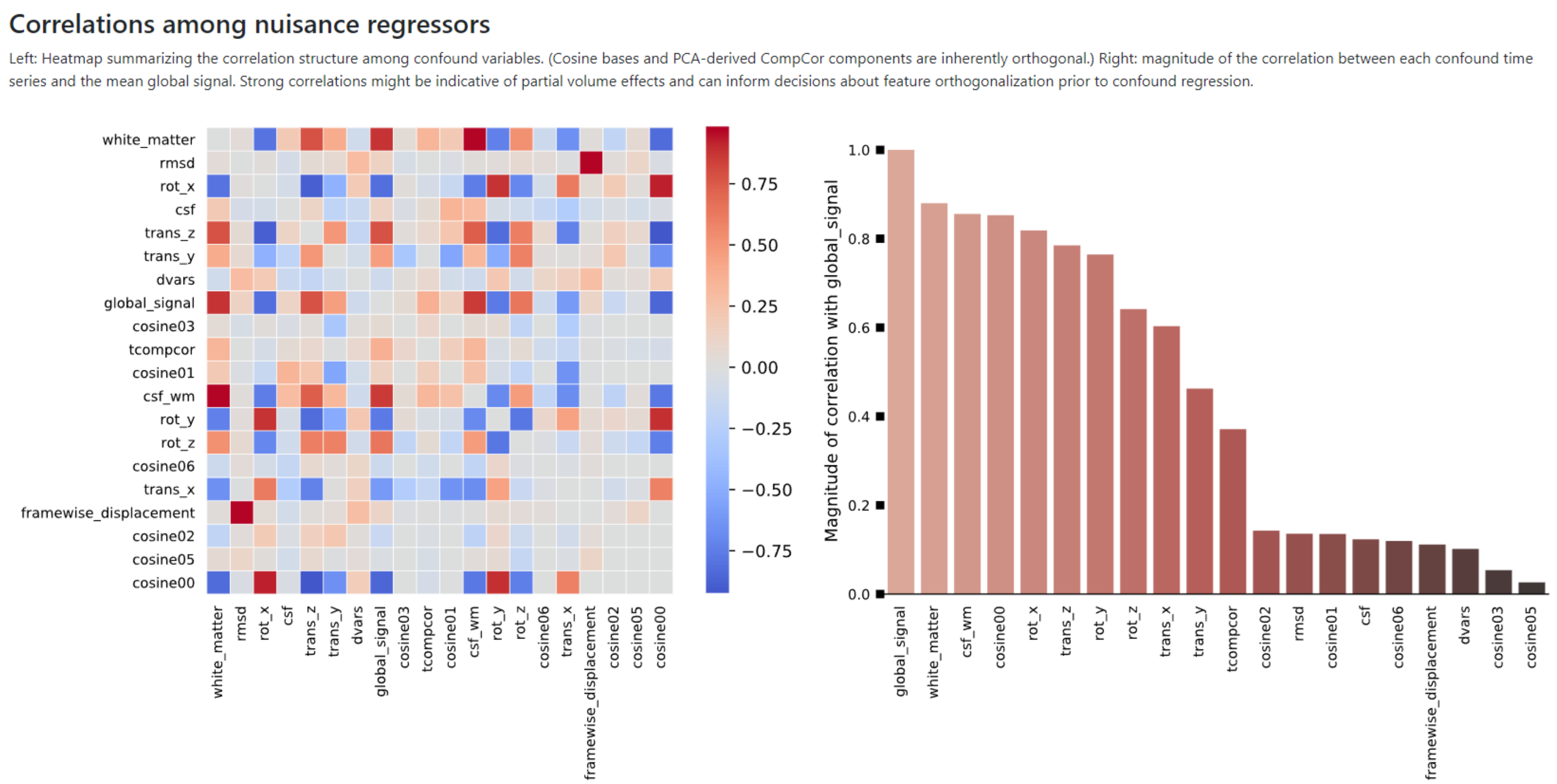
A brief overview of how to use fmriprep



Any sudden changes in motion may be reflected in uniform changes across the entire column for that timepoint.



A brief overview of how to use fmriprep



Correlations among nuisance regressors

Left: Heatmap summarizing the correlation structure among confound variables. (Cosine bases and PCA-derived CompCor components are inherently orthogonal.) Right: magnitude of the correlation between each confound time series and the mean global signal. Strong correlations might be indicative of partial volume effects and can inform decisions about feature orthogonalization prior to confound regression.

The bar chart on the right shows the correlation of different regressors with respect to global signal; those components that show a high degree of correlation may be candidates for nuisance regression.

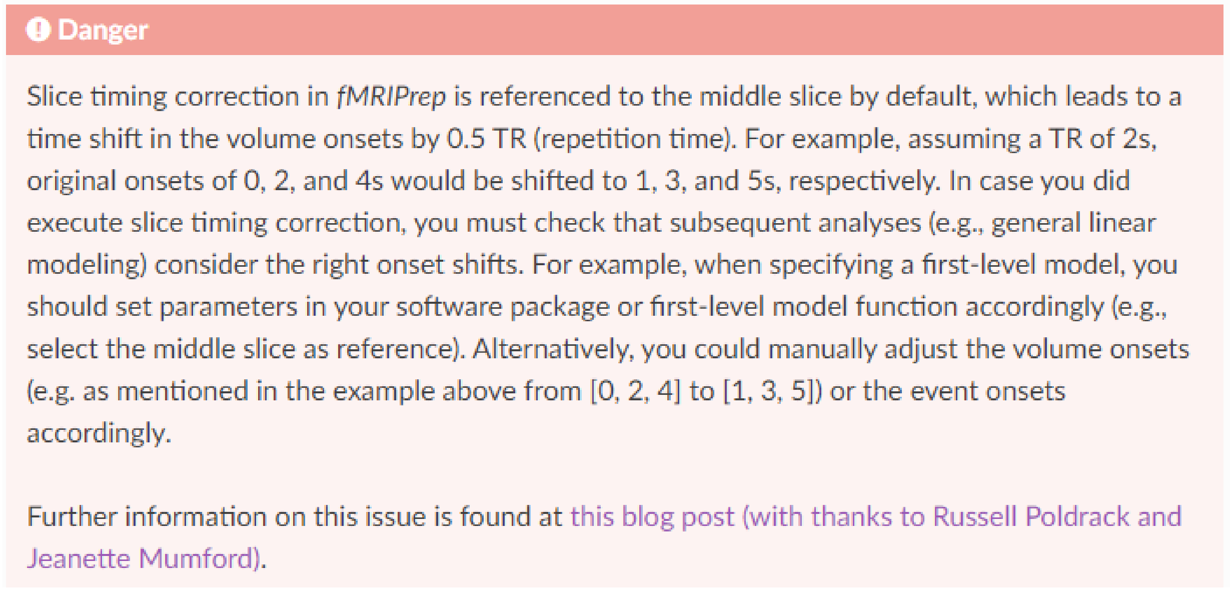
Example file:///F:/fMRIprep_HP/derivatives/fmriprep-20.2.1/sub-101.html

A brief overview of how to use fmriprep

Tips

A brief overview of how to use fmriprep

A brief overview of how to use fmriprep

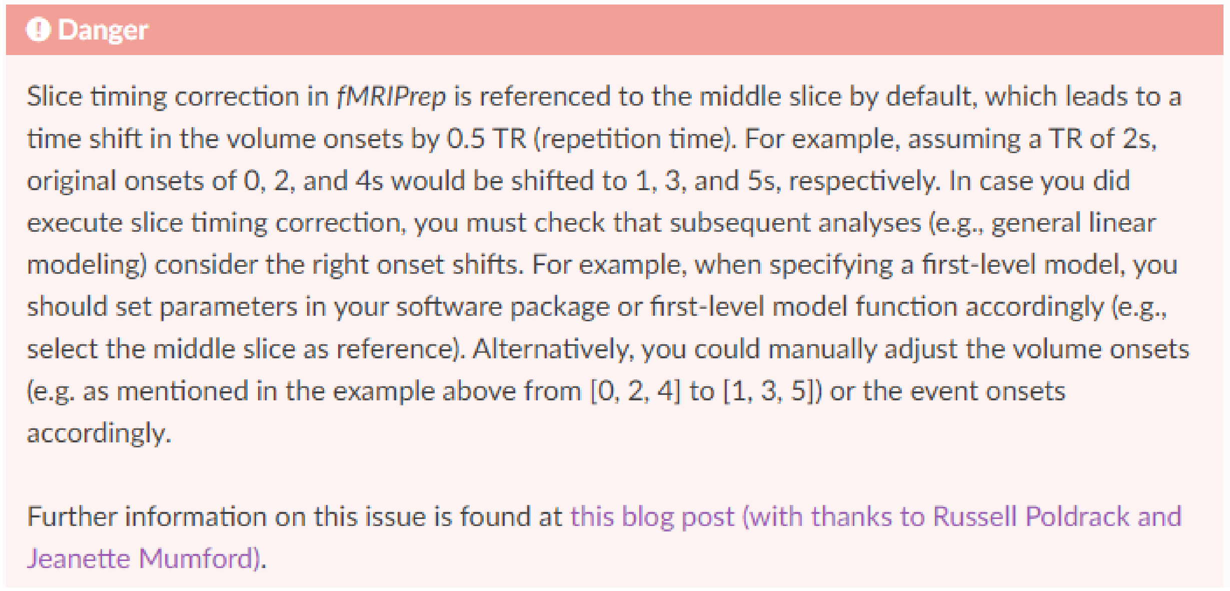


Danger

Slice timing correction in *fMRIPrep* is referenced to the middle slice by default, which leads to a time shift in the volume onsets by 0.5 TR (repetition time). For example, assuming a TR of 2s, original onsets of 0, 2, and 4s would be shifted to 1, 3, and 5s, respectively. In case you did execute slice timing correction, you must check that subsequent analyses (e.g., general linear modeling) consider the right onset shifts. For example, when specifying a first-level model, you should set parameters in your software package or first-level model function accordingly (e.g., select the middle slice as reference). Alternatively, you could manually adjust the volume onsets (e.g. as mentioned in the example above from [0, 2, 4] to [1, 3, 5]) or the event onsets accordingly.

Further information on this issue is found at this blog post (with thanks to Russell Poldrack and Jeanette Mumford).

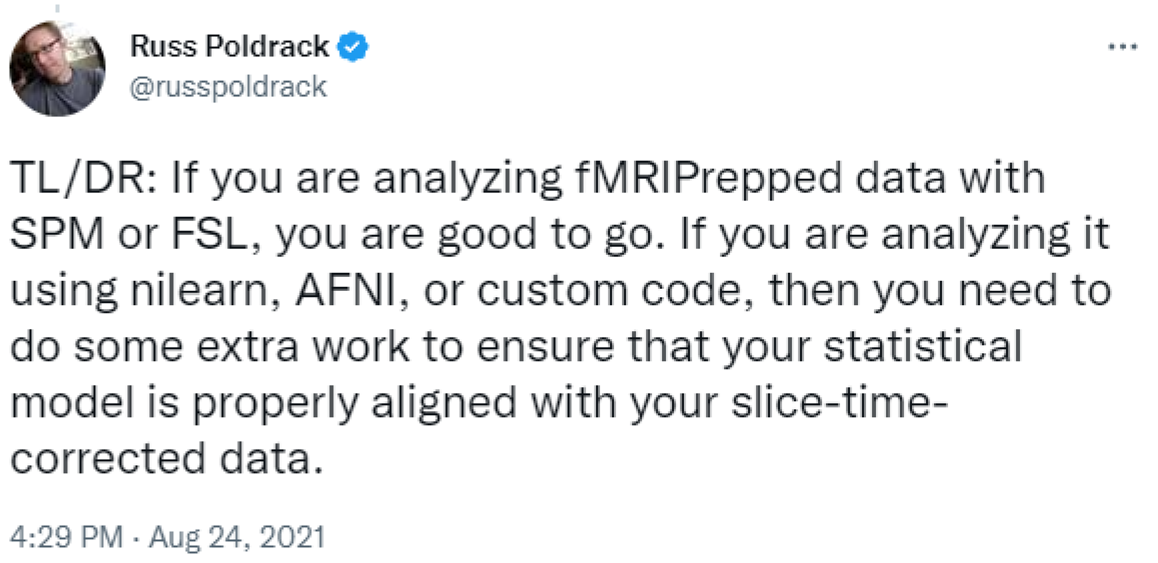
A brief overview of how to use fmriprep



Danger

Slice timing correction in *fMRIPrep* is referenced to the middle slice by default, which leads to a time shift in the volume onsets by 0.5 TR (repetition time). For example, assuming a TR of 2s, original onsets of 0, 2, and 4s would be shifted to 1, 3, and 5s, respectively. In case you did execute slice timing correction, you must check that subsequent analyses (e.g., general linear modeling) consider the right onset shifts. For example, when specifying a first-level model, you should set parameters in your software package or first-level model function accordingly (e.g., select the middle slice as reference). Alternatively, you could manually adjust the volume onsets (e.g. as mentioned in the example above from [0, 2, 4] to [1, 3, 5]) or the event onsets accordingly.

Further information on this issue is found at this blog post (with thanks to Russell Poldrack and Jeanette Mumford).

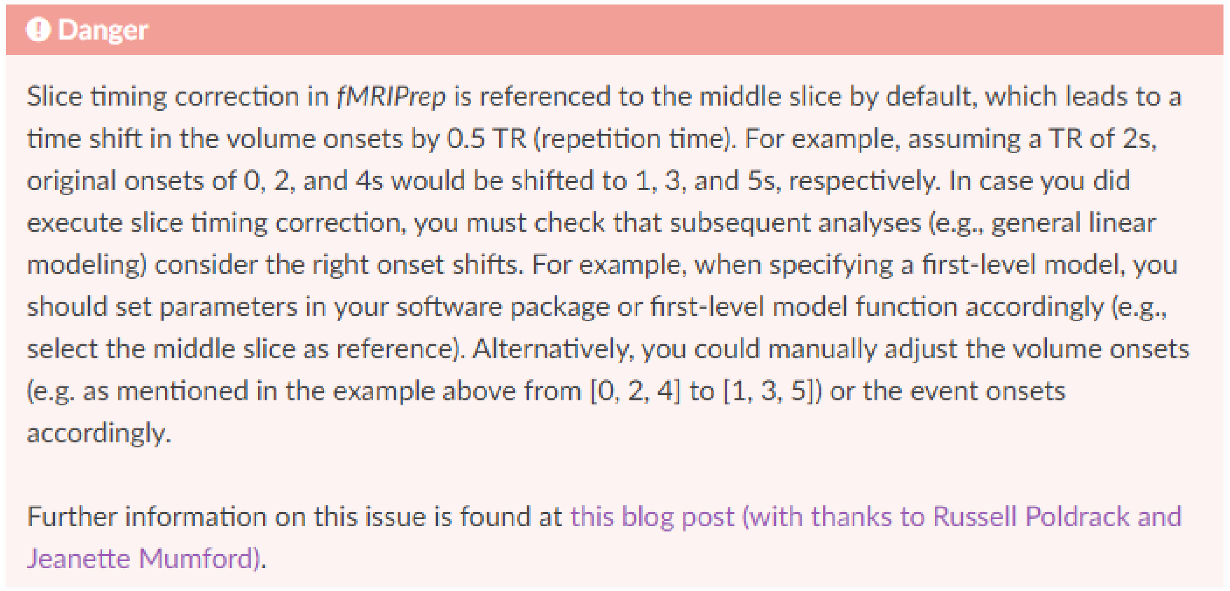


Russ Poldrack
@russpoldrack

TL/DR: If you are analyzing fMRIPrepped data with SPM or FSL, you are good to go. If you are analyzing it using nilearn, AFNI, or custom code, then you need to do some extra work to ensure that your statistical model is properly aligned with your slice-time-corrected data.

4:29 PM · Aug 24, 2021

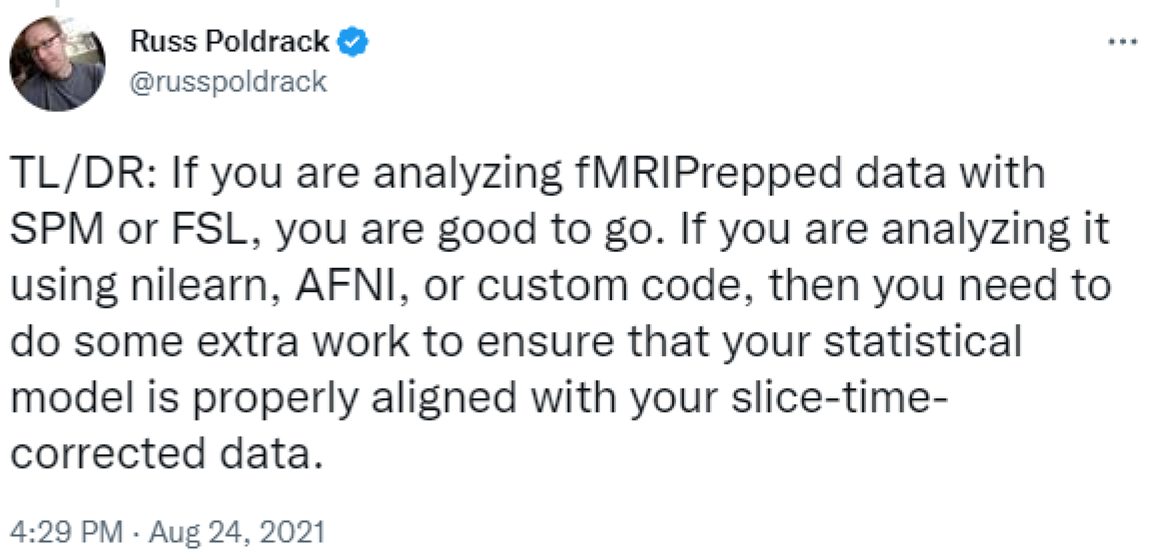
A brief overview of how to use fmriprep



Danger

Slice timing correction in *fMRIPrep* is referenced to the middle slice by default, which leads to a time shift in the volume onsets by 0.5 TR (repetition time). For example, assuming a TR of 2s, original onsets of 0, 2, and 4s would be shifted to 1, 3, and 5s, respectively. In case you did execute slice timing correction, you must check that subsequent analyses (e.g., general linear modeling) consider the right onset shifts. For example, when specifying a first-level model, you should set parameters in your software package or first-level model function accordingly (e.g., select the middle slice as reference). Alternatively, you could manually adjust the volume onsets (e.g. as mentioned in the example above from [0, 2, 4] to [1, 3, 5]) or the event onsets accordingly.

Further information on this issue is found at this blog post (with thanks to Russell Poldrack and Jeanette Mumford).



Russ Poldrack
@russpoldrack

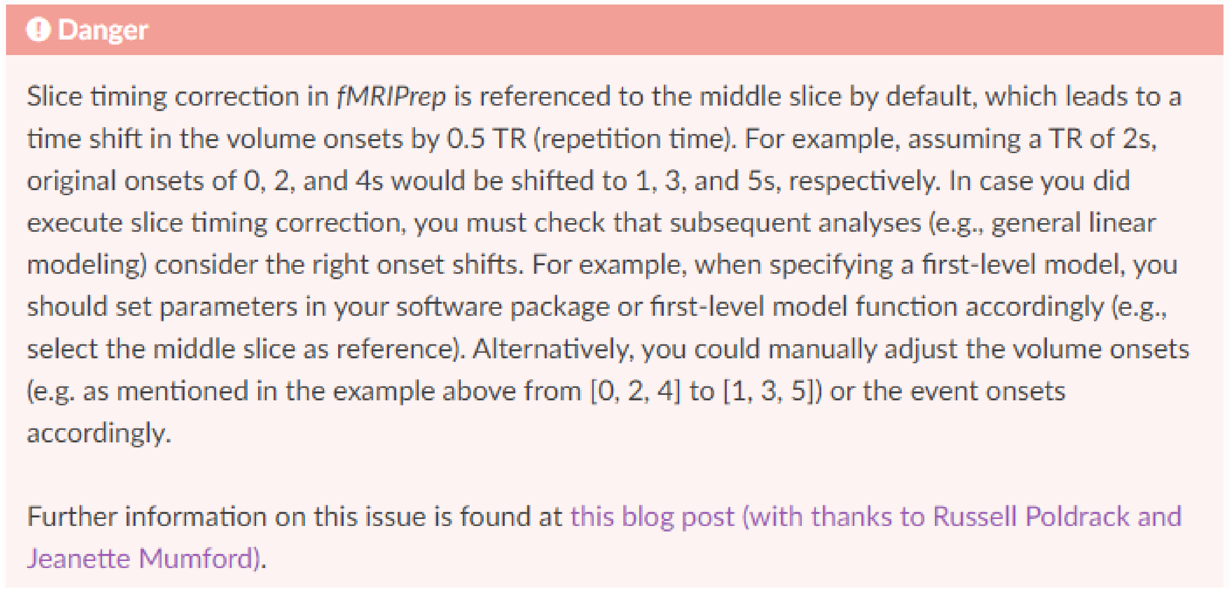
TL/DR: If you are analyzing fMRIPrepped data with SPM or FSL, you are good to go. If you are analyzing it using nilearn, AFNI, or custom code, then you need to do some extra work to ensure that your statistical model is properly aligned with your slice-time-corrected data.

4:29 PM · Aug 24, 2021

Imagine you have 45 subjects. You already preprocessed all of them, and your supervisor realized he has 6 more subjects you can add to your sample.
Then it can be possible that you get error message if you add these six subjects to your previous folder.

=> Create a new folder from scratch and launch the preprocessing again

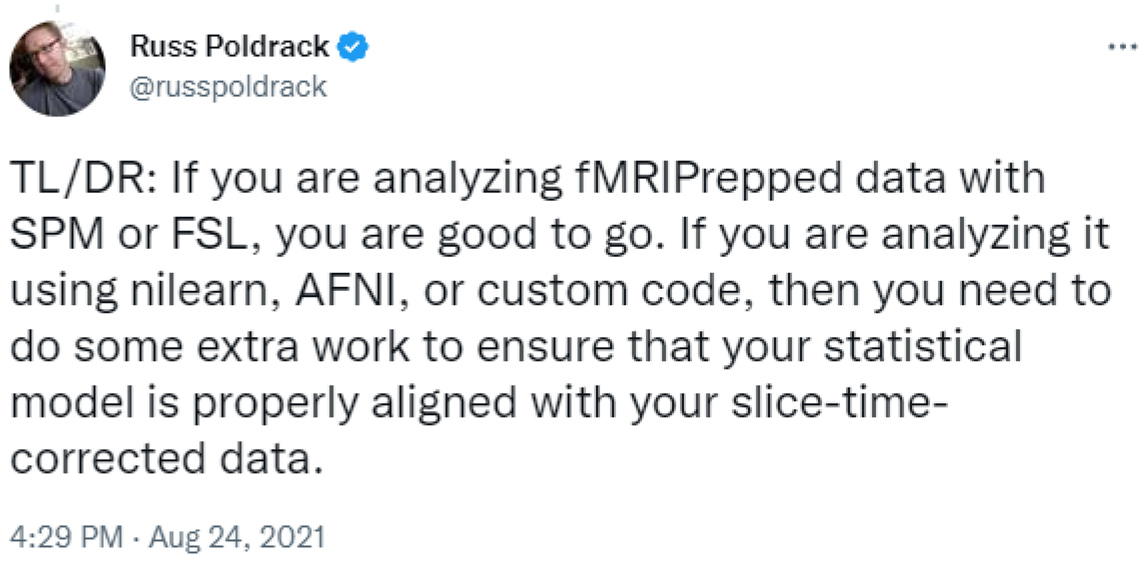
A brief overview of how to use fmriprep



Danger

Slice timing correction in *fMRIPrep* is referenced to the middle slice by default, which leads to a time shift in the volume onsets by 0.5 TR (repetition time). For example, assuming a TR of 2s, original onsets of 0, 2, and 4s would be shifted to 1, 3, and 5s, respectively. In case you did execute slice timing correction, you must check that subsequent analyses (e.g., general linear modeling) consider the right onset shifts. For example, when specifying a first-level model, you should set parameters in your software package or first-level model function accordingly (e.g., select the middle slice as reference). Alternatively, you could manually adjust the volume onsets (e.g. as mentioned in the example above from [0, 2, 4] to [1, 3, 5]) or the event onsets accordingly.

Further information on this issue is found at this blog post (with thanks to Russell Poldrack and Jeanette Mumford).



Russ Poldrack
@russpoldrack

TL/DR: If you are analyzing fMRIPrepped data with SPM or FSL, you are good to go. If you are analyzing it using nilearn, AFNI, or custom code, then you need to do some extra work to ensure that your statistical model is properly aligned with your slice-time-corrected data.

4:29 PM · Aug 24, 2021

Imagine you have 45 subjects. You already preprocessed all of them, and your supervisor realized he has 6 more subjects you can add to your sample.
Then it can be possible that you get error message if you add these six subjects to your previous folder.

=> Create a new folder from scratch and launch the preprocessing again

It take approximatively a day to process one subject.
Parallelization if recommended.

If you want to speed up the process use:
--fs-no=reconall