Preprocessing manual with fmriprep

TRAINING SYNESTHESIA "LETTER-COLOR" MRI EXPERIMENT

Claire Zhang, 23/09/2024

Ruggero Condorelli 09/12/2024

Olympia Colizoli 09/01/2025

In this manual, all commands calling modules are based on the setup of HPC cluster at DCCN. If you are using a different server, you need to install and load bidscoin and fmriprep accordingly.

Connect to the DCCN HPC cluster (Olympia's Mac):

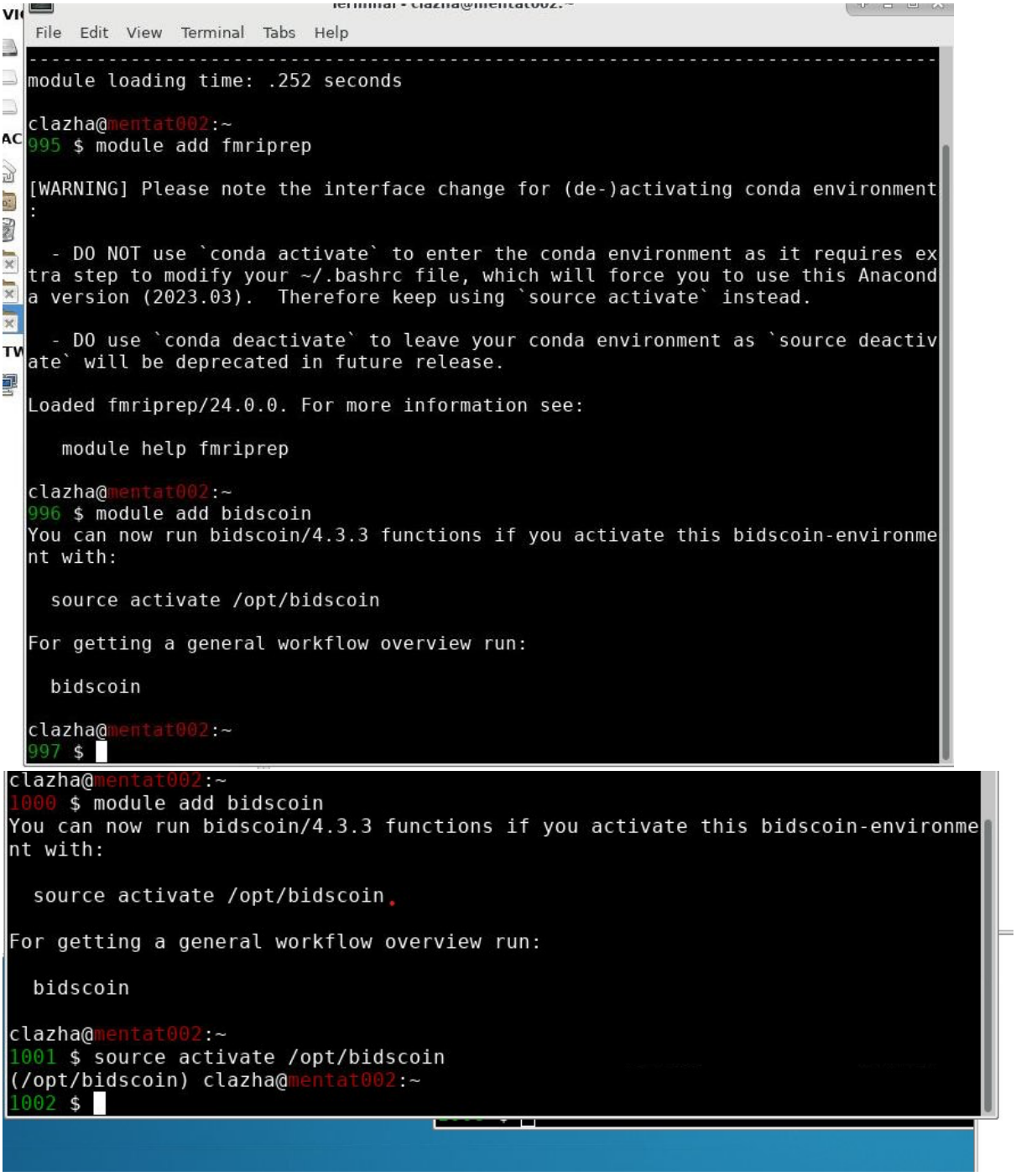
- Have to have xquartz installed on a mac for graphic mode
- The -YC flag is essential for getting GUIs to work on the mac

SSH mentat (nodes 1-5):

```
ssh -YC olycol@mentat006.dccn.nl
module load slurm
srun --x11 --time=10:00:00 --mem=4gb -p interactive --pty bash -i
```

1. Load bidscoin module:

```
module add bidscoin
source activate /opt/bidscoin
```



2. Create bidsmap with the command bidsmapper Codes:

```
bidsmapper /project/***/sourcedata /project/***/bids
```

```
bidsmapper /project/3018051.01/ruggero/raw
/project/3018051.01/ruggero/bids
```

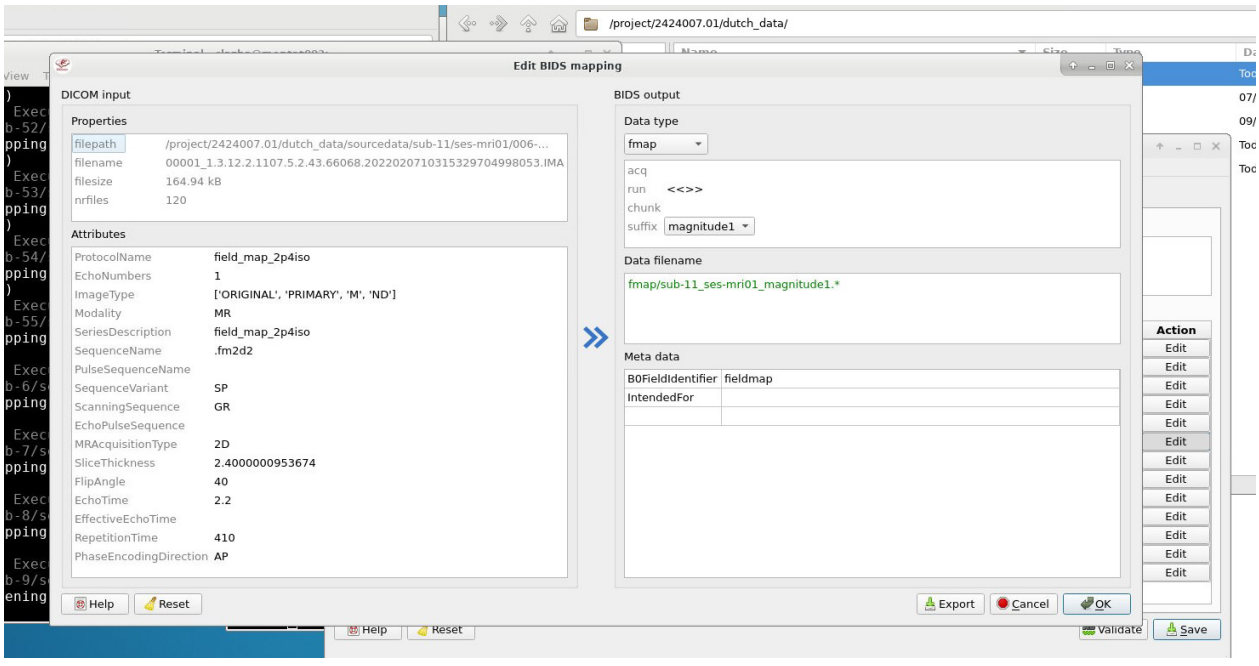
Note: first entry, should be the source data; second entry, should be the bids folder where you want to put your niifti files.

```
clazha@mentat002:~
1001 $ source activate /opt/bidscoin
(/opt/bidscoin) clazha@mentat002:~
1002 $ bidsmapper /project/2424007.01/dutch_data/sourcedata /project/2424007.01/
dutch_data/bids
INFO |
INFO | -------------- START BIDSmapper ------------
INFO | >>> bidsmapper sourcefolder=/project/2424007.01/dutch_data/sourcedata bid
sfolder=/project/2424007.01/dutch_data/bids bidsmap=bidsmap.yaml template=/home/
predatt/clazha/.bidscoin/4.3.3+qt5/templates/bidsmap_dccn.yaml plugins=[] subpre
fix=None sesprefix=None store=False force=False
INFO | No existing bidsmap file found: /project/2424007.01/dutch_data/bids/code/
bidscoin/bidsmap.yaml
INFO | Reading: /home/predatt/clazha/.bidscoin/4.3.3+qt5/templates/bidsmap_dccn.
yaml
INFO | Checking the bidsmap run-items:
SUCCESS | All datatypes and options in the template bidsmap are valid
INFO | Mapping: /project/2424007.01/dutch_data/sourcedata/sub-11/ses-mri01 (subj
ect 1/40)
VERBOSE | Executing plugin: dcm2niix2bids -> /project/2424007.01/dutch_data/sour
cedata/sub-11/ses-mri01
INFO | Discovered 'exclude' DICOM sample: /project/2424007.01/dutch_data/sourced
ata/sub-11/ses-mri01/001-localizer/00001_1.3.12.2.1107.5.2.43.66068.202202071028
2395193396288.IMA
INFO | Discovered 'exclude' DICOM sample: /project/2424007.01/dutch_data/sourced
ata/sub-11/ses-mri01/002-AAHead_Scout_32ch-head-coil/00001_1.3.12.2.1107.5.2.43.
66068.20220207102844477242965 75.IMA
INFO | Discovered 'exclude' DICOM sample: /project/2424007.01/dutch_data/sourced
ata/sub-11/ses-mri01/003-AAHead_Scout_32ch-head-coil_MPR_sag/00001_1.3.12.2.1107
.5.2.43.66068.20220207102848133914975 97.IMA
INFO | Discovered 'exclude' DICOM sample: /project/2424007.01/dutch_data/sourced
```

3. Edit bidsmap

Wait for a few seconds, then you will see this GUI. Now you can edit your json files in this GUI.

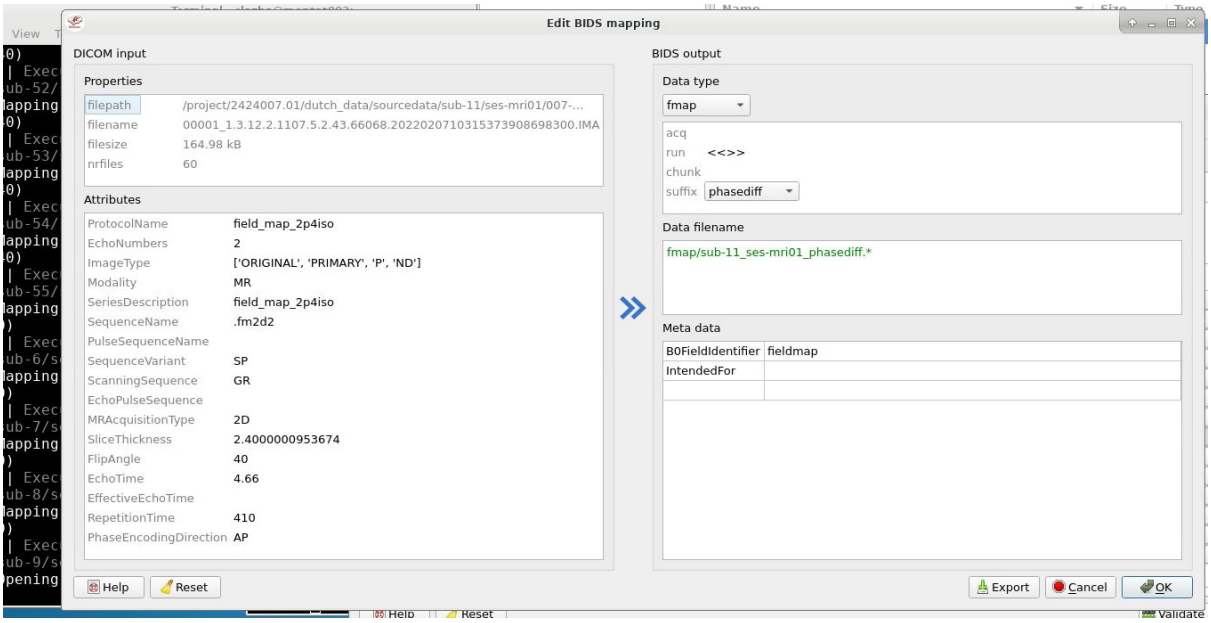
For fieldmap scans, magnitude 1 and 2:



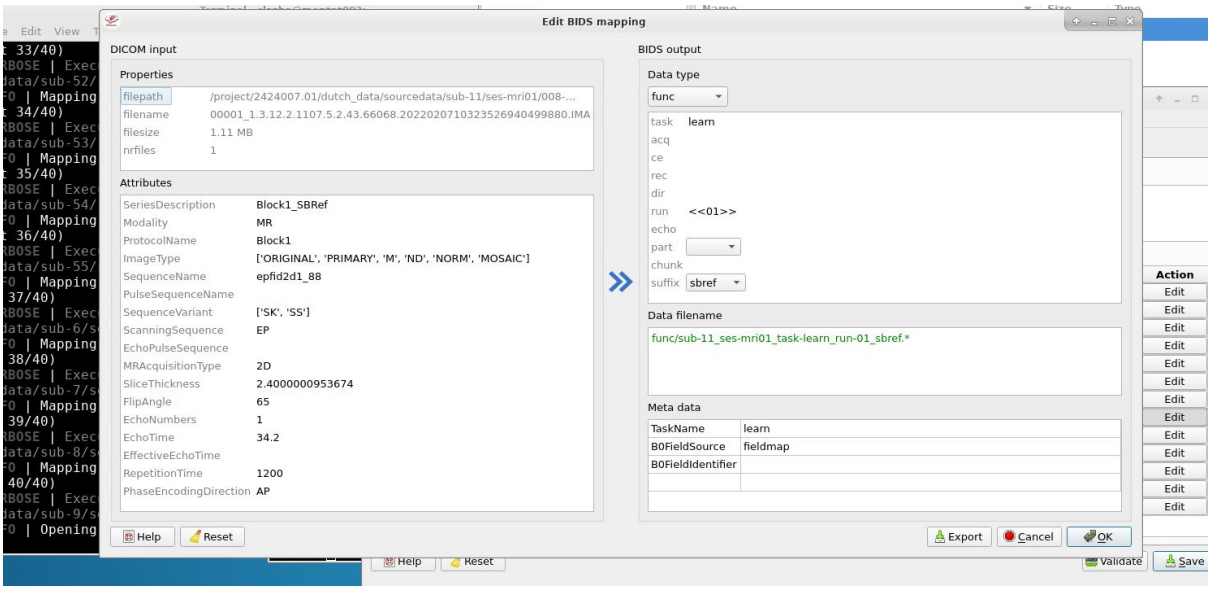
Note: Sometimes, bidsmapper only recognize 1 magnitude file. That is ok. When you run bidscoiner, it would extract magnitude 2 files as well.

This is because at DCCN, the scanner is set up to put magnitude 1 and 2 original IMA files in the same folder.

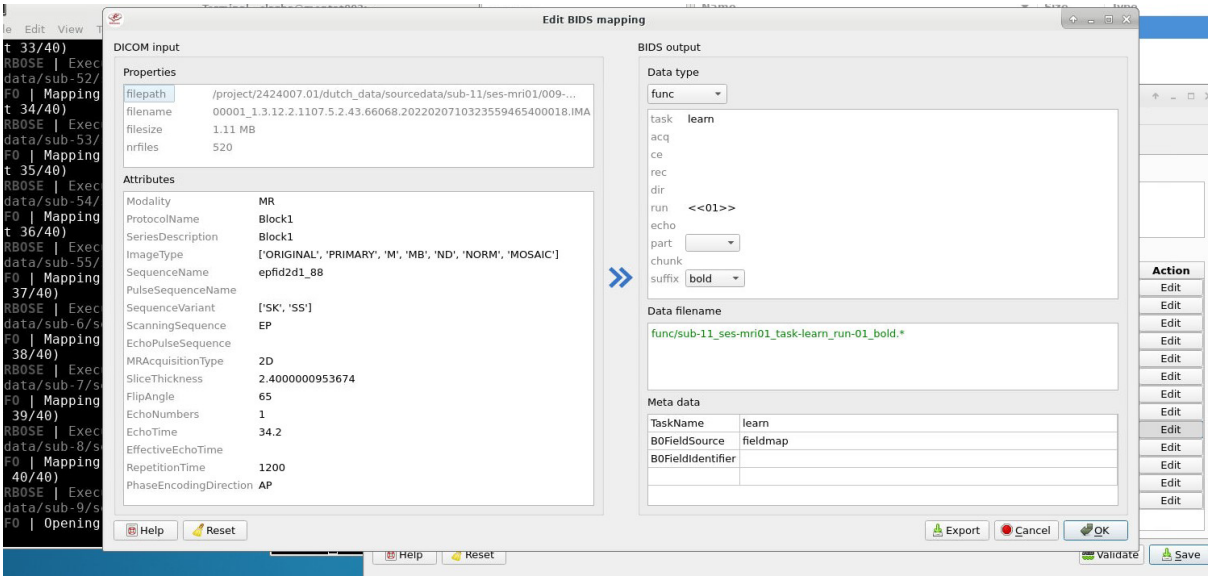
For fieldmap scans, phase difference:



For functional scans, single band:



For functional scans, bold:



IMPORTANT:

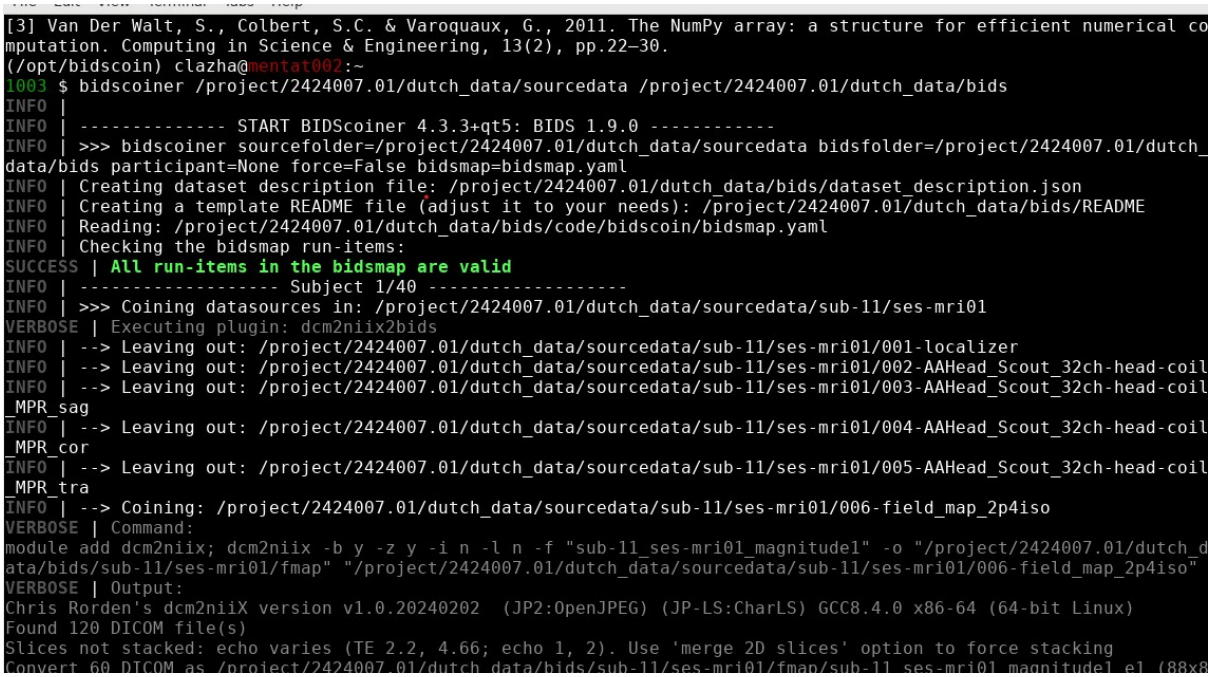
Put a key in B0FieldIdentifier in fieldmap scans. Put this same key in B0FieldSource in functional scans, both single band and bold files.

In the images, the key that used "fieldmap". This key should be the same for the magnitude 1, magnitude 2, and phase difference files for the same fieldmap. You can use anything, as long as they match with each other and it is a unique string.

If you have several fieldmaps, and you want to apply different fieldmaps to different functional scans, then you should put different keys for different fieldmaps. As long as the Source and the Identifier match with each other, fmriprep will apply them accordingly.

If you have multiple sessions, add the dynamic value "_<<session>>" to your key (e.g. if the key is "fieldmap", it should be "fieldmap_<<session>>").

This is crucial. Otherwise, fmriprep will not run fieldmap correction.



After editing, you should click on save at GUI of bidsmapper at the default folder it recommends. Usually, this map would be in this directory: /project/***/bids/code/bidscoin/. Then, close the GUI even if you see “Waiting for bidsmapper to process”. There is no notice of saying bidsmapper is saved…

4. Run bidscoiner Codes:

```
bidscoiner /project/***/sourcedata /project/***/bids
bidscoiner /project/3018051.01/ruggero/raw
/project/3018051.01/ruggero/temp_bids
```

Note: These entries should be the same as your bidsmapper command.

If you have multiple sessions, bidscoiner might not remove the arrow brackets from the fieldmap key stored in the json files.

Here’s a bash script that you can use to edit all json files (fmap, and func) and remove arrow brackets from B0FieldIdentifier and B0FieldSource tags:

```
#!/bin/bash

# Base directory to search for JSON files
BASE_DIR="/project/***/bids/"

# Find all .json files in subdirectories find "$BASE_DIR" -
type f -name "*.json" | while read -r file; do
```

```
    # Process the B0FieldIdentifier field          jq 'if
has("B0FieldIdentifier") and (.B0FieldIdentifier | type ==
"string") then .B0FieldIdentifier |= gsub("<<|>>"; "") else .
end' "$file" > "${file}.tmp" && mv "${file}.tmp" "$file"

    # Process the B0FieldSource field            jq 'if
has("B0FieldSource") and (.B0FieldSource | type == "string")
then .B0FieldSource |= gsub("<<|>>"; "") else . end' "$file" >
"${file}.tmp" && mv "${file}.tmp" "$file"

    # Check for errors     if [[ $? -eq 0 ]]; then
echo "Processed: $file"     else          echo "Error
processing: $file"     fi
done

echo "Processing complete."
```

This is to ensure that fmriprep is able to read fieldmap tags and process the fieldmaps.

5. Run fmriprep

Note: fmriprep 24.0.0 at HPC cluster is the default.

```
# use slurm
module add fmriprep
```

Codes:

24.0.0

SINGLE SUBJECT or SPECIFIC SUBJECTS

```
fmriprep_sub.py /project/3018051.01/ruggero/bids -o
/project/3018051.01/ruggero/derivatives/fmriprep –p sub-126 --nthreads
3 -r ‘slurm’ --mem_mb 28000 --args "
--output-spaces MNI152NLin6Asym anat --ignore slicetiming --workdir
```

```
/project/3018051.01/ruggero/tempdir"
```

ALL SUBJECTS

```
fmriprep_sub.py /project/3018051.01/ruggero/bids -o
/project/3018051.01/ruggero/derivatives/fmriprep --nthreads 3 -r
'slurm' --mem_mb 28000 --args
" --output-spaces MNI152NLin6Asym anat --ignore slicetiming --workdir
/project/301851.01/ruggero/tempdir"
```

Note:

For fmriprep 24.0.0 the "args" used are the following: *--output-spaces MNI152NLin6Asym -ignore slicetiming*

If you are running it at local computing cluster, put this to the fmriprep docker. You can choose to run fmriprep in participant batches. Then, you should add -p to call for specific participants.

Remove -r 'slurm' if you are still using torque.

HPC cluster code to check job status:

```
qstat -u username
```