**Introduction to the One-step General Registration and Extraction (OGRE) Pipeline**

[**https://github.com/PhilipLab/OGRE-pipeline**](https://github.com/PhilipLab/OGRE-pipeline)

**Last update: July 29, 2025**

**OGRE version: 1.0.0**

The goal of this project is to create an off-the-shelf mac preprocessing pipeline (with registration, spatial normalization, distortion correction and motion correction combined via one-step resampling) with outputs immediately analyzable with FSL FEAT. If you have questions or suggestions to improve this document, please contact [bphilip@wustl.edu](mailto:bphilip@wustl.edu).

Future releases of OGRE will use Docker containerization to simplify control, and [NITRC](https://www.nitrc.org/) code availability for long-term sharing and maintenance.

To be notified of future releases, set notifications for our GitHub repository <https://github.com/PhilipLab/OGRE-pipeline>.

OGRE workflow and table of contents:

1. [Create required files](#Create_Required_Files)
   1. [Create “scanlist.csv” for each subject](#Create_Scanlist), to describe & name the scans
   2. [*OPTIONAL*: Create “protocol.csv” for each study](#Create_Protocol), to aid automatic generation of scanlist.csv
   3. [Create .fsf files](#Create_FSF), for FSL FEAT analyses
2. [Convert dicom to nifti files](#DCM2NIIX)
3. [Review pipeline operation common principles](#PipelinePrinciples)
4. [Run structural pipeline](#PipeStruct) via **OGREstructpipeSETUP.sh**
5. [Run functional pipeline](#PipeFunc) via **OGREfMRIpipeSETUP.py**

*Note: Steps 6 will happen automatically as part of step 5 if you specified your FEAT analysis locations in step 1D*

1. [Perform your FEAT first- and higher-level analyses](#FEAT_L1)
2. [Cleanup (optional)](#Cleanup)
3. [Run structural and functional pipelines with a JSON file](#_8._Pipeline_setup)
4. [OGRE outputs](#_9._OGRE_outputs)
5. [Troubleshooting](#_10._Troubleshooting)

# 1) Creating required files

OGRE uses csv files to describe which scans were performed in which order (see sections 1A, 1B).

You will also need to create fsf files for your FSL analysis (section 1C), and optionally specify these fsf files' locations (section 1D).

To maintain flexibility, file names are never required to be any particular pattern; instead, OGRE learns their names when you pass the names as inputs to OGRE scripts. We recommend filenames that are recognizable to BIDS format (e.g. “sub-XXXX\_scanlist.csv”).

Future versions of OGRE will be able to draw this information from BIDS .json files.

## 1A. Create scanlist.csv

The “scanlist.csv” links the dicom names to nifti file outputs, and also is used by the rest of OGRE to identify file names and locations. **If you ever move your nifti files, make sure to update scanlist.csv so it remains correct**.

You can create “scanlist.csv” manually. An example is in **OGRE-pipeline/examples/sub-2025\_scanlist.csv**, and reproduced here. (The highlights indicate multiple EPIs, collected because the participant exited & reentered the scanner.)

,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/anat/sub-2025\_T1w

8,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/fmap/sub-2025\_acq-draw\_run-1\_dir-AP\_epi

9,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/fmap/sub-2025\_acq-draw\_run-1\_dir-PA\_epi

10,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-1\_sbref

11,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-1\_bold

13,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-1\_sbref

14,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-1\_bold

19,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-2\_sbref

20,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-2\_bold

22,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-2\_sbref

23,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-2\_bold

31,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/fmap/sub-2025\_acq-draw\_run-3\_dir-AP\_epi

32,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/fmap/sub-2025\_acq-draw\_run-3\_dir-PA\_epi

33,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-3\_sbref

34,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawRH\_run-3\_bold

36,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-3\_sbref

37,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-drawLH\_run-3\_bold

39,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/fmap/sub-2025\_acq-dbsi\_dir-PA\_epi

40,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/dwi/sub-2025\_acq-dbsi\_dwi

41,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-1\_sbref

42,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-1\_bold

44,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-2\_sbref

45,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-2\_bold

47,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-3\_sbref

48,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/func/sub-2025\_task-rest\_run-3\_bold

50,/Users/Shared/10\_Connectivity/raw\_data/sub-2025/anat/sub-2025\_T2w

In this example, the dicom names were just single numbers (e.g. “7”). OGRE uses the “IntendedFor” field of the field map’s JSON file to determine which field map applies to which functional scan. If the field map lacks a JSON file or if the “IntendedFor” field is not found, then OGRE will apply the most recent one to a given functional scan.

* If **OGREfMRIpipeSETUP.py** is called with the option **--donotuseIntendedFor**, then OGRE will apply the most recent field map to a given func scan.

For automatic generation of scanlists, we have included **pdf2scanlist.py** which is designed to take a XNAT screenshot pdf and compare it to the “protocol.csv” (see below) to create a “scanlist.csv”.

**% pdf2scanlist.py /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_CNDA.pdf -p /Users/mcavoy/repo/NRL-misc/10\_Connectivity\_protocol.csv**

When using pdf2scanlist.py, review your pdf’s and scanlists to ensure that no scans have been omitted (e.g. due to page breaks).

## 1B. Create protocol.csv (OPTIONAL)

The “protocol.csv“ defines the data that was acquired in the scanner, and is only required if you intend to use **pdf2scanlist.py**. If you are creating your scanlist.csv files manually, “protocol.csv” is not needed.

Consider the example for study “10\_Connectivity“:

t1\_mpr\_1mm\_p2\_pos50, overwrite, anat, T1w

SpinEchoFieldMap2\_AP, append, fmap, dir-AP\_epi

SpinEchoFieldMap2\_PA, append, fmap, dir-PA\_epi

CMRR\_fMRI\_TASK\_R1\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawRH\_run-1\_sbref

CMRR\_fMRI\_TASK\_R1\_AP\_3mm\_488meas, overwrite, func, task-drawRH\_run-1\_bold

CMRR\_fMRI\_TASK\_R2\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawLH\_run-1\_sbref

CMRR\_fMRI\_TASK\_R2\_AP\_3mm\_488meas, overwrite, func, task-drawLH\_run-1\_bold

CMRR\_fMRI\_TASK\_R3\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawRH\_run-2\_sbref

CMRR\_fMRI\_TASK\_R3\_AP\_3mm\_488meas, overwrite, func, task-drawRH\_run-2\_bold

CMRR\_fMRI\_TASK\_R4\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawLH\_run-2\_sbref

CMRR\_fMRI\_TASK\_R4\_AP\_3mm\_488meas, overwrite, func, task-drawLH\_run-2\_bold

CMRR\_fMRI\_TASK\_R5\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawRH\_run-3\_sbref

CMRR\_fMRI\_TASK\_R5\_AP\_3mm\_488meas, overwrite, func, task-drawRH\_run-3\_bold

CMRR\_fMRI\_TASK\_R6\_AP\_3mm\_488meas\_SBRef, overwrite, func, task-drawLH\_run-3\_sbref

CMRR\_fMRI\_TASK\_R6\_AP\_3mm\_488meas, overwrite, func, task-drawLH\_run-3\_bold

ep2ddbsi\_b0\_PE=PA, overwrite, fmap, acq-dbsi\_dir-PA\_epi

ep2ddbsi\_19\_2mm\_iso\_LowBW\_2av, overwrite, dwi, acq-dbsi\_dwi

CMRR\_fMRI\_REST\_R1\_AP\_3mm\_550meas\_SBRef, overwrite, func, task-rest\_run-1\_sbref

CMRR\_fMRI\_REST\_R1\_AP\_3mm\_550meas, overwrite, func, task-rest\_run-1\_bold

CMRR\_fMRI\_REST\_R2\_AP\_3mm\_550meas\_SBRef, overwrite, func, task-rest\_run-2\_sbref

CMRR\_fMRI\_REST\_R2\_AP\_3mm\_550meas, overwrite, func, task-rest\_run-2\_bold

CMRR\_fMRI\_REST\_R3\_AP\_3mm\_550meas\_SBRef, overwrite, func, task-rest\_run-3\_sbref

CMRR\_fMRI\_REST\_R3\_AP\_3mm\_550meas, overwrite, func, task-rest\_run-3\_bold

t2\_spc\_sag\_p2\_iso1.0, overwrite, anat, T2w

Each item (row) contains 4 comma-separated fields:

1. Scan name. As received from scanner.
2. Write mode: can be *append* or *overwrite*
   1. *Append*: retains all scans with this name. See highlighted lines in examples.
   2. *Overwrite*: if there are multiple scans with this exact name, retains only the last one.
3. Bids directory: specify a subdirectory for BIDS compliance. E.g. *anat*, *fmap*, *func*, *dwi*
4. Output root: BIDS-compliant filename without extension.

## 1C. Create .fsf files for FSL FEAT analysis

This document assumes you are already familiar with FSL. FSL documentation is available [here](https://fsl.fmrib.ox.ac.uk/fsl/docs/#/).

*If this is your first time running OGRE, we recommend saving this until after you've successfully run the functional pipeline. That* way you will have the pipeline outputs on hand. (You are never required to create your .fsf files in advance; you will run your FEAT analysis at the end of the functional pipeline.)

To use FSL FEAT on data preprocessed by OGRE, you will use FEAT's “statistics” option (rather than “full analysis”), to skip preprocessing and registration.

Confirm in your first-level .fsf files: “set fmri(analysis) 2”

In addition, FEAT will always try to run SUSAN spatial smoothing, even if you turn off preprocessing. Therefore, you must also set the spatial smoothing kernel to 0 in your first-level analyses. You can do this in the GUI "Preprocessing" tab if you temporarily switch back to "full analysis," but the important check is:

Confirm in your first-level .fsf files: “set fmri(smooth) 0”

*Alternatively, you could perform your spatial smoothing (if any) in FEAT instead of OGRE. We find that logistically cumbersome, but it's functionally identical - FSL and OGRE use the same tools. The critical part is to avoid accidentally smoothing your data more times than you intended!*

You will also need to select the appropriate outputs of OGRE’s functional script to serve as your inputs to the first-level FEAT analysis. These will be found in **/my/project/directory/func**. For example, if your project directory is

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/func**

You will choose

If no smoothing: **sub-2025\_task-drawLH\_run-1\_OGRE-preproc\_bold.nii.gz**  
If using SUSAN smoothing from **-f** (**--fwhm**): **sub-2025\_task-drawLH\_run-1\_OGRE-preproc\_susan-#mm\_hptf-%s\_bold.nii.gz**

*(Where # is your SUSAN smoothing FWHM, and % is your temporal cutoff)*

*Ex. sub-2025\_task-drawLH\_run-1\_OGRE-preproc\_susan-6mm\_hptf-60s\_bold.nii.gz*

Otherwise, create your .fsf files as normal for your study design. Example first- and second-level .fsf's are included in OGRE-pipeline/examples.

Note that if you wish to include motion parameters in your model, you will need to do so via the "additional confound EVs" option. A text file of standard MCFLIRT motion parameters can be found in OGRE output at **/my/project/directory/regressors**. For example:

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/regressors/sub-2025\_task-drawLH\_run-1\_bold\_mc.par**

A good practice is to put all your FEAT in **/my/project/directory/sub-XXXX\_model-OGRE**. For example:

**/Users/Shared/10\_Connectivity/derivatives/analysis/sub-2025/sub-2025\_model-OGRE**

# 2. Convert dicoms to niftis

OGREdcm2niix.sh uses Chris Rorden’s dcm2niix to perform the dicom to nifti conversion automatically on an entire scanlist.csv.

**% OGREdcm2niix.sh /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv \**

**-i /Volumes/NRLbackup/10\_Connectivity/dicom/sub-2025**

In the above example, the “scanlist.csv“ created in section 1A is used to convert the dicoms located at **/Volumes/NRLbackup/10\_Connectivity/dicom/sub-2025** as specified by the -i option.

If nifti files are not created, check **/Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_dcm2niix.sh.txt**. If it says “input folder invalid“ this may be a permissions error, run chmod -R 775 or 777 on the subject directory.

**% chmod -R 775 /Users/Shared/10\_Connectivity/raw\_data/sub-2025**

Note that this step is “required” in the sense that nifti files are required for OGRE. You may be able to use preexisting nifti files, though OGREdcm2niix.sh will ensure the creation of nifti files that match your scanlist.csv.

For compatibility with non-OGRE software, OGREdcm2niix.sh adds the "IntendedFor" field in each JSON to link your functional scans to the most recent fieldmap in the scanlist (if any). OGRE in turn reads this field to know which field maps are applied to which functional scans. Thus there are two ways a user may indicate which field maps are applied to a given functional scan. The first is in the scanlist.csv, and the second is by editing the “IntendedFor” field in the functional scan’s JSON.

# 3. Pipeline operation common principles

The easiest way to use OGRE is to:

1) Create a JSON that defines the pipeline setup.

2) Call **OGREsetup.py** with the JSON file as an argument.

**% OGREsetup.py sub-2025.json**

Both the structural and functional pipeline setup scripts are called, and an executable “\_fileout.sh” is created that runs the entire pipeline. Simply invoke from a command prompt.

**% /Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_OGREbatch\_fileout.sh**

To provide a thorough understanding, we will first explain how to use the pipeline with explicit calls to the structural and functional setup scripts as this will clarify how the arguments are coded in the JSON file.

The main OGRE scripts are **OGREstructpipeSETUP.sh** and **OGREfMRIpipeSETUP.py** that take a “scanlist.csv” and other arguments to create subject-specific bash scripts. Subject-specific scripts are always accompanied by a “\_fileout.sh” variant. The difference is that the “\_fileout.sh” version will send its output to a text file instead of the command line. *The real processing happens when you run the subject-specific scripts.*

The setup scripts work in the project directory whose default location is calculated from the *location* of your “scanlist.csv”.

1) Following BIDS nomenclature, if your “scanlist.csv” is in **parent/raw\_data/sub-XXXX**, then the project directory is

**parent/derivatives/preprocessed/sub-XXXX**

2) If your “scanlist.csv” isn’t in **parent/raw\_data/sub-XXXX**, then the project directory is the *location* of your “scanlist.csv”.

To override this default behavior, use the **-P** (**--projectdirectory**) argument to explicitly specify where the output should be created (for the structural pipeline) and found (for the functional pipeline).

% **OGREstructpipeSETUP.sh** */path/to/***sub-2025\_scanlist.csv -P** */my/project/directory*

% **OGREfMRIpipeSETUP.py** */path/to/***sub-2025\_scanlist.csv --P** */my/project/directory*

After calling **OGREstructpipeSETUP.sh**, within the project directory a **pipeline”Freesurfer version#”** subdirectory is created. For the remainder of this tutorial we will assume that Freesurfer version 7.4.1 is used (i.e. **pipeline7.4.1**), but the pipeline has also been tested with Freesurfer versions 5.3.0-HCP, 7.2.0, 7.3.2 and 7.4.0 which can be set with **--freesurferversion**. Within **pipeline7.4.1** a **scripts** folder is created where the OGRE scripts are written, and a **templates** folder where the structural templates are copied. When the pipeline is run, additional folders are created that hold the interim outputs. Final outputs will appear in separate BIDS subdirectories (e.g. **anat**, **func** and **regressors**) alongside the **pipeline7.4.1** subdirectory.

To automatically execute the “\_fileout.sh” at the moment when you create it via a SETUP script, use the **--autorun** argument when calling the SETUP script.

% **OGREstructpipeSETUP.sh sub-2025\_scanlist.csv --autorun**

% **OGREfMRIpipeSETUP.py sub-2025\_scanlist.csv --autorun**

A convenient way to set up the structural and functional pipelines together is via the **--batchscript** option.

**% OGREstructpipeSETUP.sh sub-2025\_scanlist.csv [arguments] --batchscript**

**% OGREfMRIpipeSETUP.py sub-2025\_scanlist.csv [arguments] --batchscript/ --autorun**

This will create:

**% /Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_OGREbatch\_fileout.sh**

which is identical to running the structural script, followed by the functional script. Note that **-A** option results in automatic execution of the “fileout.sh” and, with the sequential setup calls, is only used as an argument to **OGREfMRIpipeSETUP.py**.

*Do not execute the functional pipeline's subject-specific scripts until the structural pipeline has completed.*

In the description of each pipeline it mentions "uses [scantype] (if available)." OGRE will find & use these optional files if they are included in scanlist.csv.

# 4. Structural pipeline

The structural pipeline uses the T1 (required) and T2 (if available) to extract, segment, parcellate, and register the brain along with spatial normalization.

Basic usage is with “scanlist.csv“ as the sole argument,

**% export OGREDIR=/path/to/OGRE-pipeline**

**% OGREstructpipeSETUP.sh /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv**

or with the **--OGREDIR** option

% **OGREstructpipeSETUP.sh /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv \**

**-OGREDIR /path/to/OGRE-pipeline**

The use of the option will override OGREDIR set with an export declaration.

This will create the scripts

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_OGREstruct.sh**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_OGREstruct\_fileout.sh**

As described in the Common Principles, the processing happens when you run the "struct.sh" script (for output to command line) or "struct\_fileout.sh" (for output to a text file).

The structural setup script will also copy your scanlist.csv file to the scripts folder,

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/scripts**

and the structural templates to the templates folder,

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/templates**

The default T1-weighted and T2-weighted (optional) templates are the the MNI152 asymmetric (HCP/FSL version) templates with a high resolution of 1 mm and low resolution of 2 mm. However, user chosen templates can be specified by providing a path to the location of the high resolution templates

**--ht /my/path/to/the/high/res/templates**

and/or a path the low resolution templates

**--lt /my/path/to/the/low/res/templates**

Template files may also specified individually, see

**% OGREstructpipeSETUP.sh --helpall**

Templates specified individually have precedence over those found through the use of the **--highres-template** and **--lowres-template** arguments. As state above, the templates are copied to the **templates** folder where they will be accessed by the functional pipeline.

# 5. Functional pipeline

The functional pipeline uses the outputs of the structural pipeline along with the SBRef images (if available) and field maps (if available) to preprocess the BOLD time series. The implementation follows that of the structural pipeline.

**% export OGREDIR=/path/to/OGRE-pipeline**

**% OGREfMRIpipeSETUP.py /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv**

This will create the scripts

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_OGREfMRIvol.sh**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_OGREfMRIvol\_fileout.sh**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_bidscp.sh**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_smooth.sh**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/sub-2025\_cleanup.sh**

The main processing happens when you run the "fMRIvol.sh" (for output to command line) or "fMRIvol\_fileout.sh" (for output to a text file). As mentioned previously, the latter calls "fMRIvol.sh".

The helper scripts “bidscp.sh”, “smooth.sh” and “featadapter.sh” are called from “OGREfMRIvol.sh” but can also be run as standalone scripts:

**“bidscp.sh“** copies the OGRE pipeline outputs in **/my/project/directory/pipeline7.4.1** to the appropriate BIDS directories, e.g

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/func**

**/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/regressors**

In addition, BIDS-compliant JSON metadata files are produced alongside the output files in **/my/project/directory/func**.

**“smooth.sh”** performs SUSAN spatial smoothing and FSL’s (nonlinear) temporal high and low pass filtering. Even if smoothing is not specified, a template “smooth.sh” script is written which you may edit and run. e.g.

**--fwhm 6** *6 mm FWHM SUSAN spatial smoothing*

high pass filtering with the cutoff specified in seconds or Hz,

**--hpf\_sec 60** *High pass filter cutoff of 60 s*

**--hpf\_Hz 0.0167** *High pass filter cutoff of 0.0167 Hz ~ 60 s*

and low pass filtering in seconds or Hz,

**--lpf\_sec 12.5** *Low pass filter cutoff of 12.5 s*

**--lpf\_Hz 0.08** *Los pass filter cutoff of 0.08 Hz = 12.5 s*

**“featadapter.sh”** runs first and second level FEAT analyses. These are specified with **--feat**, e.g.

**--feat /my/path/myfile.fsf** *Single fsf file*

**--feat /path/to/fsf/files** *Directory with fsf files*

**--feat feat\_locator.txt** *Text file which lists fsf files and/or directories with fsf files*

**--feat /my/path/myfile.fsf /path/to/fsf/files feat\_locator.txt** *All three of the above*

As illustrated above, the **--feat** argument allows you to specifiy a combination of individual files, paths to files and text lists (i.e. *locator* files) of individual files and/or paths to files. “fsf” files may include a combination of first and second level FEAT analyses. An additional standalone script “featadapter.sh” is created that is called by “fMRIvol.sh”.

*If this is your first time running OGRE, we recommend omitting -feat. Then you can use the outputs to set up your “fsf” files, and thereafter run the FEAT adapter with the* ***--featadapter*** *argument as shown below.*

**% OGREfMRIpipeSETUP.py /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv \**

**--feat /my/path/myfile.fsf /path/to/fsf/files feat\_locator.txt --featadapter**

In the *locator* files, the individual files and/or paths to files can be separated by commas, spaces, tabs and/or newlines. See **OGRE-pipeline/examples/sub-2025\_locatorOne.txt** and **OGRE-pipeline/examples/sub-2025\_locatorTwo.txt**. The FEAT adapter is described below in step 6.

**“cleanup.sh”** is also a standalone script that removes the interim files created by the pipeline, however it is not called by “OGREfMRIvol.sh”. *We advise against calling “cleanup.sh” until you have checked the output files.* The "cleanup.sh" script is described in step 7.

Normally you will not need to invoke “bidscp.sh,” “smooth.sh,” or “featadapter.sh;” these are automatically executed by “fMRIvol.sh.” However, OGRE’s modular design allows you to easily modify and run these standalone scripts independently.

# 6. FEAT adapter

One additional step is needed between first-level FEAT analysis and higher-level FEAT analyses. The first-level results will lack registration information, because FEAT does not (yet) know about all the registration work completed by OGRE. The “featadapter.sh” script will copy the appropriate transformations and matrices to create a “reg“ directory inside each of the first-level FEAT output directories. All of this is automatically performed by “featadapter.sh” which is executed by “fMRIvol.sh”, but you have the power to edit “featadapter.sh” and run manually as it is a standalone script.

% **sub-2025\_featadapter.sh**

As noted above, “featadapter.sh” is automatically created as part of the functional pipeline. To recreate “featadapter.sh” without running any other analysis, use the --**featadapter** option. For example,

**% OGREfMRIpipeSETUP.py /Users/Shared/10\_Connectivity/raw\_data/sub-2025/sub-2025\_scanlist.csv --feat sub-2025\_locatorOne.txt sub-2025\_locatorTwo.txt --featadapter**

This will create a **sub-XXXX\_featadapter\_fileout.sh** and **sub-XXXX\_featadapter.sh**; run either of those (as always, fileout doesn't lock up your terminal window).

You may find it useful to run the adapter manually on individual .feat directories. The script that does the work here is **OGRE-pipeline/lib/OGREmakeregdir.sh**, which is documented and ready for independent use on an individual .feat directory:

% **OGREmakeregdir.sh /my/path/sub-2025\_task-drawLH\_run-1.feat**

This is the same as running “featadapter.sh” on a single directory. One of the things “featadapter.sh” does is run **OGREmakeregdir.sh** on each of the .feat directories you provide.

# 7. Cleanup

If you are satisfied that you know which files you need, you can save disk space by deleting everything except the final BIDS-format outputs via "cleanup.sh", e.g.:

% **/Users/Shared/10\_Connectivity/derivatives/preprocessed/sub-2025/pipeline7.4.1/scripts/sub-2025\_cleanup.sh**

would delete all OGRE subdirectories in the directory **derivatives/preprocessed/pipeline7.4.1** except **scripts**, **templates**, **MNINonLinear/Results** and **MNINonLinear/gm\_wm\_csf**. Individual files in the project directory and non-OGRE subdirectories are retained.

The only reason to retain **pipeline7.4.1** files is if you want to use them in your analyses. These are most likely found in **MNINonLinear/Results** and **MNINonLinear/gm\_wm\_csf**. For any others, you should of course copy out any useful files before running “cleanup.sh”. You may wish to edit “cleanup.sh” to suit your needs.

If you need to rerun “struct.sh”, it is easiest to delete all existing outputs first. If you need to rerun “fMRIvol.sh”, it will overwrite any previous functional outputs.

# 8. Pipeline setup with a JSON file

As mentioned previously, with a JSON file, the entire pipeline can be setup and run with a single call.

**% OGREsetup.py /my/path/to/json/sub-2025.json**

The default JSON file is found at

**OGRE-pipeline/lib/PY/opl/OGREdefault.json**

In the JSON, you only need to specify fields that change from the default. A brief explanation of the currently recognized fields follows.

**{**

**"OGREstructpipeSETUP": true, true:** *setup the structural script* **false:** *don’t setup the structural script*

**"OGREfMRIpipeSETUP": true, true:** *setup the functional script* **false:** *don’t setup the functional script*

**"ScanList": null,** *Ex. /my/path/to*/sub-2025\_scanlist.csv

*Equivalent to* **-s** (**-ScanList**) *option.*

**"AutoRun": false, true:** *run pipeline* **false:** *don’t run pipeline*

*Equivalent to* **-A** (**-AutoRun**) *option.*

**"OGREDIR": null,** *Setting it here will override your environment setting. Ex. /my/path/to/OGRE-pipline*

*Equivalent to* **-O** (**-OGREDIR**) *option.*

**"HCPDIR": null,** *Only need to set if different from the default location /my/path/to/OGRE-pipline/lib/HCP*

*Equivalent to* **-H** (**-HCPDIR**) *option.*

**"FreeSurferVersion": null,** *Only need to set if different from the default 7.4.1. Other versions tested include 5.3.0-HCP, 7.2.0, 7.3.2, 7.4.0.*

*Equivalent to* **-V** (**-FreeSurferVersion**) *option.*

**"HostName": false,** *If true, append machine name to pipeline directory.*

*Equivalent to* **-m** (**-HostName**) *option.*

**"Date": false,** *If true, add date (YYMMDD) to name of output script.*

*Equivalent to* **-D** (**-Date**) *option.*

**"DateLong": false,** *If true, add date (YYMMDDHHMMSS) to name of output script.*

*Equivalent to* **-L** (**-DateLong**) *option.*

**"BatchScript": true,** *“fileout.sh” scripts are collected in an executable batchscript.*

*If a filename is provided, then in addition, the batchscripts are written to the provided filename.*

*This facilitates the creation of an across-subjects script such that multiple subjects can be run sequentially and seamlessly.*

*Equivalent to* **-b** (**-BatchScript**) *option.*

**"Dilation": null,** *Only need to set if different from the default of 3 dilations of the brain mask.*

*Equivalent to* **-d** (**-Dilation**) *option for* *OGREstructpipeSETUP.sh.*

**"Erosion": null,** *Only need to set if different from the default of 2 erosions of the brain mask.*

*Equivalent to* **-e** (**-Erosion**) *option for* *OGREstructpipeSETUP.sh.*

**"HighResolutionTemplateDirectory": null,** *Location of high resolution templates. Overrides default MNI152 1mm asymmetric (HCP/FSL).*

*Equivalent to* **-ht** (**-HighResolutionTemplateDirectory**) *option for* *OGREstructpipeSETUP.sh.*

**"LowResolutionTemplateDirectory": null,** *Location of low resolution templates. Overrides default MNI152 2mm asymmetric (HCP/FSL).*

*Equivalent to* **-lt** (**-LowResolutionTemplateDirectory**) *option for OGREstructpipeSETUP.sh.*

**"ProjectDirectory": null,** *All pipeline products and working files are created inside this directory. Overrides the default location.*

*Equivalent to* **-P** (**-ProjectDirectory**) *option.*

**"Name": null,** *Use with* **ProjectDirectory** *to provide the subject name. Default is root of “scanlist.csv”. Overrides the default.*

*Equivalent to* **-n** (**-Name**) *option.*

**"HighResolution": null,** *High resolution in mm: 0.7, 0.8 or 1. Default is 1. Applies only to default MNI152 asymmetric (HCP/FSL) templates.*

*Equivalent to* **-r** (**-HighResolution**) *option for OGREstructpipeSETUP.sh.*

**"T1HighResolutionWholeHead": null,** *Default is MNI152\_T1\_1mm asymmetric (HCP/FSL). Overrides* **HighResolutionTemplateDirectory***.*

*Equivalent to* **-T1HighResolutionWholeHead** *option for OGREstructpipeSETUP.sh.*

**"T1HighResolutionBrainOnly": null,** *Default is MNI152\_T1\_1mm\_brain asymmetric (HCP/FSL). Overrides* **HighResolutionTemplateDirectory***.*

*Equivalent to* **-T1HighResolutionBrainOnly** *option for OGREstructpipeSETUP.sh.*

**"T1HighResolutionBrainMask": null,** *Default is MNI152\_T1\_1mm\_brain\_mask asymmetric (HCP/FSL). Overrides* **HighResolutionTemplateDirectory***.*

*Equivalent to* **-T1HighResolutionBrainMask** *option for OGREstructpipeSETUP.sh.*

**"T1LowResolutionWholeHead": null,** *Default is MNI152\_T1\_2mm asymmetric (HCP/FSL). Overrides* **LowResolutionTemplateDirectory***.*

*Equivalent to* **-T1LowResolutionWholeHead** *option for OGREstructpipeSETUP.sh.*

**"T1LowResolutionBrainOnly": null,** *No default. If provided, this will be used to make the low resolution mask. Overrides* **LowResolutionTemplateDirectory***.*

*Equivalent to* **-T1LowResolutionBrainOnly** *option for OGREstructpipeSETUP.sh.*

**"T1LowResolutionBrainMask": null,** *Default is MNI152\_T1\_2mm\_brain\_mask\_dil asymmetric (HCP/FSL). Overrides* **LowResolutionTemplateDirectory***.*

*Equivalent to* **-T1LowResolutionBrainMask** *option for OGREstructpipeSETUP.sh.*

*Mask is dilated if name does not include "dil".*

**"T2HighResolutionWholeHead": null,** *Default is MNI152\_T2\_1mm asymmetric (HCP/FSL). Overrides* **HighResolutionTemplateDirectory***.*

*Equivalent to* **-T2HighResolutionWholeHead** *option for OGREstructpipeSETUP.sh.*

**"T2HighResolutionBrainOnly": null,** *Default is MNI152\_T2\_1mm\_brain asymmetric (HCP/FSL). Overrides* **HighResolutionTemplateDirectory***.*

*Equivalent to* **-T2HighResolutionBrainOnly** *option for OGREstructpipeSETUP.sh.*

**"T2HighResolutionBrainMask": null,** *No default. If provided, this will be used to make* **T2HighResolutionBrainOnly***. Overrides* **HighResolutionTemplateDirectory***.*

*Equivalent to* **-T2HighResolutionBrainMask** *option for OGREstructpipeSETUP.sh.*

**"T2LowResolutionWholeHead": null,** *Default is MNI152\_T2\_2mm asymmetric (HCP/FSL). Overrides* **LowResolutionTemplateDirectory***.*

*Equivalent to* **-T2LowResolutionWholeHead** *option for OGREstructpipeSETUP.sh.*

**"FWHM": null,** *SUSAN spatial smoothing (mm). Multiple values ok.*

*Equivalent to* **-f** (**-FWHM**) *option for OGREfMRIpipeSETUP.sh.*

**"HPFcutoff\_sec": null,** *High pass filter cutoff in seconds.*

*Equivalent to* **-p** (**-HPFcutoff\_sec**) *option for OGREfMRIpipeSETUP.sh.*

**"LPFcutoff\_sec": null,** *Low pass filter cutoff in seconds.*

*Equivalent to* **-LPFcutoff\_sec** *option for OGREfMRIpipeSETUP.sh.*

**"HPFcutoff\_Hz": null,** *High pass filter cutoff in Hz.*

*Equivalent to* **-HPFcutoff\_Hz** *option for OGREfMRIpipeSETUP.sh.*

**"LPFcutoff\_Hz": null,** *Low pass filter cutoff in Hz.*

*Equivalent to* **-LPFcutoff\_Hz** *option for OGREfMRIpipeSETUP.sh.*

**"SmoothOnly": false,** *If true, only do SUSAN smoothing and low/high pass filtering if specified.*

*Equivalent to* **-SmoothOnly** *option for OGREfMRIpipeSETUP.sh.*

**"donotsmoothrest": false,** *If true, do not do SUSAN smoothing and low/high pass filtering on* **rest** *runs.*

*Equivalent to* **-donotsmoothrest** *option for OGREfMRIpipeSETUP.sh.*

**"donotuseIntendedFor": false,** *If true, do not use the* **IntendedFor** *field of the field map JSON to determine which bolds the fieldmap should be applied to. Instead use the most recent fieldmap in the scanlist.*

*Equivalent to* **-donotuseIntendedFor** *option for OGREfMRIpipeSETUP.sh.*

**"Feat": null,** *Path to fsf files, text file which lists fsf files or directories with fsf files, one or more fsf files, or a combination thereof. An OGREfeat.sh call is created for each fsf.*

*Equivalent to* **-Feat** *option for OGREfMRIpipeSETUP.sh.*

**"FeatAdapter": false,** *If true, only write the feat adapter scripts.*

*Equivalent to* **-F** (**-FeatAdapter**) *option for OGREfMRIpipeSETUP.sh.*

**"UseRefinement": false,** *If true, use the freesurfer refinement in the warp for one step resampling. Defaut is not use the refinement as this was found to misregister the bolds.*

**"fsl\_motion\_outliers": null,** *Run* **fsl\_motion\_outliers** *on raw data, with optional comma-separated arguments.*

*Equivalent to* **-fsl\_motion\_outliers** *option for OGREfMRIpipeSETUP.sh.*

*Ex.* **-- fsl\_motion\_outliers --fd,--thresh=2**

**"startIntensityNormalization": false** *If true, start at IntensityNormalization. Defaut is to run the whole OGREfMRIpipeSETUP.py script.*

*Equivalent to* **-startIntensityNormalization** *option for OGREfMRIpipeSETUP.sh.*

**}**

To make things a little easier for the user, the field keys are case-insenstive. For example, the key “ScanList” is equivalent to “scanlist”, “SCANLIST” and “sCaNLisT”.

# 9. OGRE outputs

OGRE's primary outputs are BIDS-compatible, created when “bidscp.sh” copies the OGRE pipeline's outputs from the **pipeline7.4.1** working directory to the bids output directories (*anat*, *func* and *regressors*) which reside at the same level as **pipeline7.4.1** in the project directory.

Let *project* = derivatives/preprocessed/sub-XXXX

Let *runname* = everything between the subject ID and the suffix in scanlist.csv (e.g. “task-X\_run-Y”)

Functionals:

BOLDs

From *project*/pipeline7.4.1/MNINonLinear/Results*/runname*/sub-XXXX\_*runname*\_bold.nii.gz

To *project*/func/sub-XXXX\_*runname*\_OGRE-preproc\_bold.nii.gz

Ex. **derivatives/preprocessed/sub-2025/func/sub-2025\_task-drawLH\_run-1\_OGRE-preproc\_bold.n.gz**

Brain masks

From *project*/pipeline7.4.1/MNINonLinear/Results/*runname*/brainmask\_fs.2.nii.gz

To *project*/func/sub-XXXX\_*runname*\_OGRE-preproc\_res-2\_label-brain\_mask.nii.gz

Ex. **derivatives/preprocessed/sub-2025/func/sub-2025\_task-drawLH\_run-1\_OGRE-preproc\_res-2\_label-brain\_mask.nii.gz**

Structurals:

T1 and T2 weighted whole head images

From *project*/pipeline7.4.1/MNINonLinear/TXw\_restore.nii.gz

To *project*/anat/sub-XXXX\_OGRE-preproc\_desc-restore\_TXw.nii.gz

Ex. **derivatives/preprocessed/sub-2025/anat/sub-2025\_OGRE-preproc\_desc-restore\_T1w.nii.gz**

T1 and T2 weighted brain-only images *(not BIDS-compliant, instead in “\_brain” filename format required by FSL)*

From *project*/pipeline7.4.1/MNINonLinear/TXw\_restore\_brain.nii.gz

To *project*/anat/sub-XXXX\_OGRE-preproc\_desc-restore\_TXw\_brain.nii.gz

Ex. **derivatives/preprocessed/sub-2025/anat/sub-2025\_OGRE-preproc\_desc-restore\_T1w\_brain.nii.gz**

Regressors:

MCFLIRT motion outputs (xyz translation, xyz rotation) – **In CONN, use this as “realignment”**

From: *project***/**pipeline7.4.1/sub-XXXX\_*runname*\_bold/MotionCorrection/sub-XXXX\_runname\_bold\_mc.par

To: *project*/regressors/sub-XXXX\_*runname*\_bold\_mc.par

Ex. **derivatives/preprocessed/sub-2025/regressors/sub-2025\_task-drawLH\_run-1\_bold\_mc.par**

Extended motion parameters (xyz translation, xyz rotation, derivative of xyz translation, derivative of xyz rotation)

From: *project***/**pipeline7.4.1/sub-XXXX\_*runname*\_bold/Movement\_Regressors.txt

To: *project*/regressors/sub-XXXX\_task-*runname*\_bold\_mc-withderiv.txt

Ex. **derivatives/preprocessed/sub-2025/regressors/sub-2025\_task-drawLH\_run-1\_bold\_mc-withderiv.txt**

Detrended (zero mean, zero slope) extended motion parameters.

From: *project***/**pipeline7.4.1/sub-XXXX\_task-*runname*\_bold/Movement\_Regressors\_dt.txt

To: *project*/regressors/sub-XXXX\_task-*runname*\_bold\_mc-withdetrendderiv.txt

Ex. **derivatives/preprocessed/sub-2025/regressors/sub-2025\_task-drawLH\_run-1\_bold\_mc-** **withdetrendderiv.txt**

fsl\_motion\_outliers output confound file [fsl\_motion\_outliers -o] (if any) – **In CONN, use this as “scrubbing”**

In: *project*/regressors/sub-XXXX\_task-*runname*\_bold\_fmospikes.txt

Ex. **derivatives/preprocessed/sub-2025/regressors/sub-2025\_task-drawLH\_run-1\_bold\_** **fmospikes.txt**

fsl\_motion\_outliers metric values [fsl\_motion\_outliers -s] (if any) **– In CONN, use this to determine “MeanMotion”**

In: *project*/regressors/sub-XXXX\_task-*runname*\_bold\_fmovalues.txt

Ex. **derivatives/preprocessed/sub-2025/regressors/sub-2025\_task-drawLH\_run-1\_bold\_** **fmovalues.txt**

Combined MCFLIRT motion outputs (.par above) + fsl\_motion\_outliers output (if any). If you didn't call fsl\_motion\_outliers, this is identical to the .par output. – **In FSL, use this as “additional confound EVs”**

In: *project*/regressors/sub-XXXX\_task-*runname*\_bold\_mc\_confoundevs.txt

Ex. **derivatives/preprocessed/sub-2025/regressors/sub-2025\_task-drawLH\_run-1\_bold\_** **mc\_ confoundevs.txt**

Extended motion parameters (xyz translation, xyz rotation, derivative of xyz translation, derivative of xyz rotation) and + fsl\_motion\_outliers output (if any).

In: *project*/regressors/sub-XXXX\_task-*runname*\_bold\_mc-withderiv\_confoundevs.txt

Ex. **derivatives/preprocessed/sub-2025/regressors/sub-2025\_task-drawLH\_run-1\_bold\_mc-withderiv\_confoundevs.txt**

Detrended (zero mean, zero slope) extended motion parameters + fsl\_motion\_outliers output (if any).

In: *project*/regressors/sub-XXXX\_task-*runname*\_bold\_mc-withderiv\_confoundevs.txt

Ex. **derivatives/preprocessed/sub-2025/regressors/sub-2025\_task-drawLH\_run-1\_bold\_mc-withdetrendderiv\_confoundevs.txt**

Motion outlier quality assurance file. Includes metrics for mean motion (MeanMotion), maximum motion (MaxMotion), number of invalid scans (InvalidScans), number of valid scans (ValidScans), and number of valid scans divided by the total number of scans (ProbValidScans).

In: *project*/regressors/sub-XXXX\_task-*runname*\_bold\_fmoQA.json

Ex. **derivatives/preprocessed/sub-2025/regressors/sub-2025\_task-drawLH\_run-1\_bold\_** **fmoQA.json**

Other non-BIDS outputs in MNINonLinear you may find useful:

Parcelation/segmentation masks of gray matter, white matter, & csf for T1 (via lib/OGRESplitFreeSurferMasks.sh)

In: *project***/**pipeline7.4.1/MNINonLinear/gm\_wm\_csf

Ex. **derivatives/preprocessed/sub-2025/pipeline7.4.1/MNINonLinear/gm\_wm\_csf/sub-2025\_graymatter.nii.gz**

**derivatives/preprocessed/sub-2025/pipeline7.4.1/MNINonLinear/gm\_wm\_csf/sub-2025\_whitematter.nii.gz**

**derivatives/preprocessed/sub-2025/pipeline7.4.1/MNINonLinear/gm\_wm\_csf/sub-2025\_cerebrospinalfluid.nii.gz**

# 10. Troubleshooting

Some issues and solutions we have encountered...

1. OGRE assumes the location of your “scanlist.csv” is the project directory. If this is not the case, then resolve this with **-P** (**--ProjectDirectory**).

Ex. **OGREstructpipeSETUP.sh** */path/to/***sub-0001\_scanlist.csv -P** */put/the/project/here*

1. If python scripts don't work, try the following:

**% cd OGRE-pipeline/lib/PY**

**% bash setup.sh**

1. If you need to update/recreate metadata, use:

**% OGRE-pipeline/scripts/scanlist2json.py**

1. If your "scanlist.csv" is ever incorrect, everything will crash. Therefore, if you move files/directories, you should update "scanlist.csv."
   1. When you run the struct pipeline, it copies your current scanlist to the pipeline7.4.1 directory. Beware of versioning errors if you update one copy but use the other.
2. If you rerun the struct pipeline (“struct.sh”) for a subject, you should first delete any preexisting outputs from any previous run of the struct pipeline (complete or partial).