CMP3 Documentation

Release v3.1.0

Brain Communication Pathways Sinergia Consortium

Oct 25, 2022

GETTING STARTED

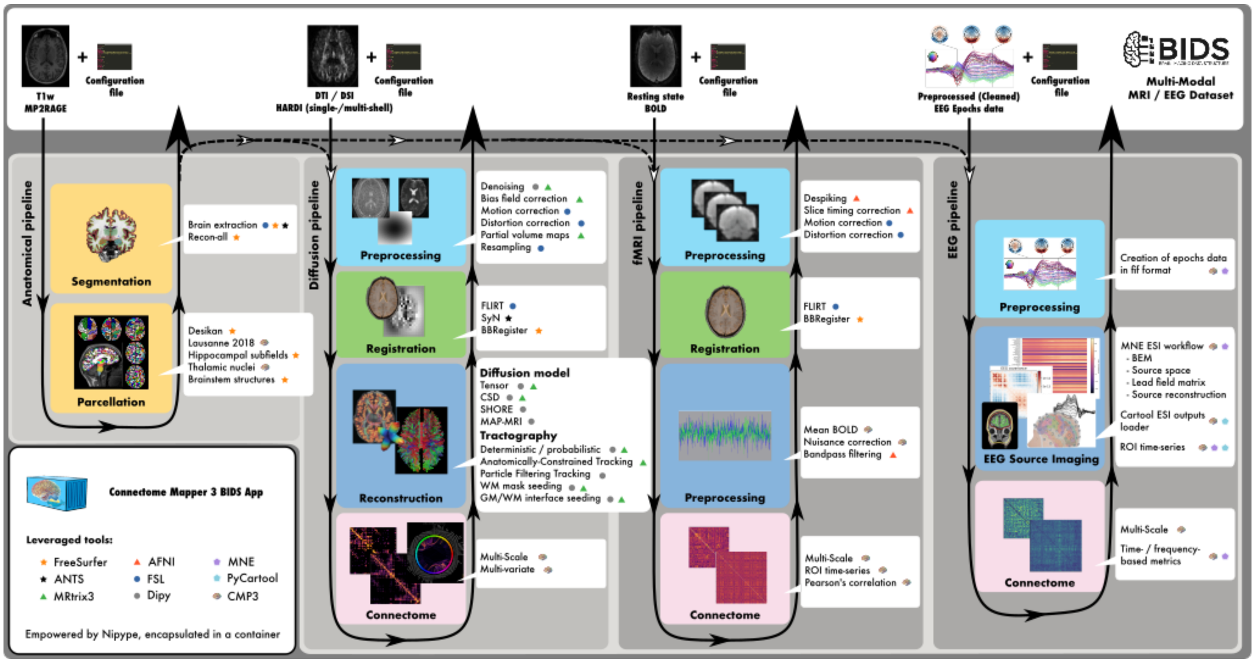
Latest released version: v3.1.0

This neuroimaging processing pipeline software is developed by the Connectomics Lab at the University Hospital of Lausanne (CHUV) for use within the SNF Sinergia Project 170873, as well as for open-source software distribution. Source code is hosted on GitHub.

> **Warning:** THIS SOFTWARE IS FOR RESEARCH PURPOSES ONLY AND SHALL NOT BE USED FOR ANY CLINICAL USE. THIS SOFTWARE HAS NOT BEEN REVIEWED OR APPROVED BY THE FOOD AND DRUG ADMINISTRATION OR EQUIVALENT AUTHORITY, AND IS FOR NON-CLINICAL, IRB-APPROVED RESEARCH USE ONLY. IN NO EVENT SHALL DATA OR IMAGES GENERATED THROUGH THE USE OF THE SOFTWARE BE USED IN THE PROVISION OF PATIENT CARE.

CHAPTER ONE

ABOUT



Connectome Mapper 3 is an open-source Python3 image processing pipeline software, with a Graphical User Interface (GUI), that implements full anatomical, diffusion and resting-state MRI processing pipelines, from raw T1 / Diffusion / BOLD / preprocessed EEG data to multi-resolution connection matrices based on a new version of the Lausanne parcellation atlas, aka `Lausanne2018`.

`Connectome Mapper 3` pipelines use a combination of tools from well-known software packages, including FSL, FreeSurfer, ANTs, MRtrix3, Dipy, AFNI, MNE, MNEcon, and PyCartool empowered by the Nipype dataflow library. These pipelines are designed to provide the best software implementation for each state of processing at the time of conception, and can be easily updated as newer and better neuroimaging software become available.

To enhance reproducibility and replicatibility, the processing pipelines with all dependencies are encapsulated in a Docker image container, which handles datasets organized following the BIDS standard and is distributed as a BIDS App @ Docker Hub. For execution on high-performance computing cluster, a Singularity image is also made freely available @ Sylabs Cloud.

To enhanced accessibility and reduce the risk of misconfiguration, Connectome Mapper 3 comes with an interactive GUI, aka `cmpbidsappmanager`, which supports the user in all the steps involved in the configuration of the pipelines, the configuration and execution of the BIDS App, and the control of the output quality. In addition, to facilitate the use by users not familiar with Docker and Singularity containers, Connectome Mapper 3 provides two Python commandline wrappers (`connectomemapper3_docker` and `connectomemapper3_singularity`) that will generate and run the appropriate command.

Since `v3.1.0`, CMP3 provides full support to EEG. Please check this notebook for a demonstration using the public VEPCON dataset.

Carbon footprint estimation of BIDS App run

In support to the Organisation for Human Brain Mapping (OHBM) Sustainability and Environmental Action (OHBM-SEA) group, CMP3 enables you since `v3.0.3` to be more aware about the adverse impact of your processing on the environment!

With the new `--track_carbon_footprint` option of the `connectomemapper3_docker` and `connectomemapper3_singularity` BIDS App python wrappers, and the new `"Track carbon footprint"` option of the BIDS Interface Window of `cmpbidsappmanager`, you can estimate the carbon footprint incurred by the execution of the BIDS App. Estimations are conducted using codecarbon to estimate the amount of carbon dioxide (CO2) produced to execute the code by the computing resources and save the results in `<bids_dir>/code/emissions.csv`.

Then, to visualize, interpret and track the evolution of the emitted CO2 emissions, you can use the visualization tool of `codecarbon` aka `carbonboard` that takes as input the `csv` created:

```
$ carbonboard --filepath="<bids_dir>/code/emissions.csv" --port=xxxx
```

Please check https://ohbm-environment.org to learn more about OHBM-SEA!

CHAPTER TWO

LICENSE INFORMATION

This software is distributed under the open-source license Modified BSD. See *license* for more details.

All trademarks referenced herein are property of their respective holders.

CHAPTER

THREE

AKNOWLEDGMENT

If your are using the Connectome Mapper 3 in your work, please acknowledge this software. See *Citing* for more details.

3.1 Help/Questions

If you run into any problems or have any questions, you can post to the CMTK-users group. Code bugs can be reported by creating a "New Issue" on the source code repository.

CHAPTER

FOUR

EAGER TO CONTRIBUTE?

Connectome Mapper 3 is open-source and all kind of contributions (bug reporting, documentation, code,…) are welcome! See *Contributing to Connectome Mapper* for more details.

CHAPTER

FIVE

CONTENTS

5.1 Installation Instructions for Users

Warning: This software is for research purposes only and shall not be used for any clinical use. This software has not been reviewed or approved by the Food and Drug Administration or equivalent authority, and is for non-clinical, IRB-approved Research Use Only. In no event shall data or images generated through the use of the Software be used in the provision of patient care.

The Connectome Mapper 3 is composed of a Docker image, namely the Connectome Mapper 3 BIDS App, and a Python Graphical User Interface, namely the Connectome Mapper BIDS App Manager.

- Installation instructions for the Connectome mapper 3 BIDS App are found in *Installation*.
- Installation instructions for the Connectome mapper 3 BIDS App Manager are found in *Installation*.

Make sure that you have installed the following prerequisites.

Important: On Mac and Windows, if you want to track the carbon emission incurred by the processing with the `--track_carbon_footprint` option flag, you will need to install in addition the `Intel Power Gadget` tool available here.

5.1.1 The Connectome Mapper 3 BIDSApp

Prerequisites

- Install Docker Engine depending of your system:
  - For Ubuntu 14.04/16.04/18.04, follow the instructions at https://docs.docker.com/install/linux/docker-ce/ubuntu/
  - For Mac OSX (>=10.10.3), get the .dmg installer at https://store.docker.com/editions/community/docker-ce-desktop-mac
  - For Windows (>=10), get the installer at https://store.docker.com/editions/community/docker-ce-desktop-windows

Note: Connectome Mapper 3 BIDSApp has been tested only on Ubuntu and MacOSX. In principles, it should also run on Windows but it might require a few patches to make it work.

- Manage Docker as a non-root user
  - Open a terminal
  - Create the docker group:

    ```
    $ sudo groupadd docker
    ```

  - Add the current user to the docker group:

    ```
    $ sudo usermod -G docker -a $USER
    ```

  - Reboot
  - After reboot, test if docker is managed as non-root:

    ```
    $ docker run hello-world
    ```

Installation

Installation of the Connectome Mapper 3 has been facilitated through the distribution of a BIDSApp relying on the Docker software container technology.

- Open a terminal
- Download and extract the latest release (v3.1.0) of the BIDS App:

```
$ docker pull sebastientourbier/connectomemapper-bidsapp:v3.1.0
```

Note: This can take some time depending on your connection speed and your machine. The docker image of the BIDSApp has a compressed size of 6.28 GB on DockerHub and should take 17.6 GB of space on your machine after download and extraction.

- To display all docker images available:

  ```
  $ docker images
  ```

  You should see the docker image "connectomemapper-bidsapp" with tag "v3.1.0" is now available.

- You are ready to use the Connectome Mapper 3 BIDS App from the terminal. See its commandline usage.

5.1.2 The Connectome Mapper 3 BIDSApp Manager (GUI)

Prerequisites

- Download the Python 3 installer of miniconda3 corresponding to your 32/64bits MacOSX/Linux/Win system and install it following the instructions at https://conda.io/miniconda.html.

Installation

The installation of the Connectome Mapper 3, including `cmpbidsappmanager`, consists of the creation of conda environment with all python dependencies installed, and the installation of `connectomemapper` via the Python Package Index (PyPI) as follows:

- Download the appropriate environment.yml / environment_macosx.yml.

  Important: It seems there is no conda package for `git-annex` available on Mac. For your convenience, we created an additional `conda/environment_macosx.yml` miniconda3 environment where the line `- git-annex=XXXXXXX` has been removed. Git-annex should be installed on MacOSX using brew i.e. `brew install git-annex`. See https://git-annex.branchable.com/install/ for more details.

  Note that `git-annex` is only necessary if you wish to use BIDS datasets managed by Datalad (https://www.datalad.org/).

- Open a terminal.
- Create a miniconda3 environment where all python dependencies will be installed:

```
$ conda env create -f /path/to/downloaded/conda/environment[_macosx].yml
```

  Note: This can take some time depending on your connection speed and your machine. It should take around 2.8GB of space on your machine.

- Activate the conda environment:

```
$ source activate py39cmp-gui
```

  or:

```
$ conda activate py39cmp-gui
```

- Install finally the latest released version of Connectome Mapper 3 with the Python Package Index (PyPI) using `pip`:

```
(py39cmp-gui)$ pip install connectomemapper
```

- You are ready to use the Connectome Mapper 3 (1) via its Graphical User Interface (GUI) aka CMP BIDS App Manager (See *Graphical User Interface* for the user guide), (2) via its python `connectomemapper3_docker` and `connectomemapper3_singularity` wrappers (See *With the wrappers* for commandline usage), or (3) by interacting directly with the Docker / Singularity Engine (See *With the Docker / Singularity Engine* for commandline usage).

In the future

If you wish to update Connectome Mapper 3 and the Connectome Mapper 3 BIDS App Manager, this could be easily done by running `pip install connectomemapper==v3.X.Y`.

Help/Questions

If you run into any problems or have any questions, you can post to the CMTK-users group. Code bugs can be reported by creating a "New Issue" on the source code repository.

5.2 Connectome Mapper 3 and the BIDS standard

`Connectome Mapper 3 (CMP3)` adopts the BIDS (Brain Imaging Data Structure) standard for data organization and is developed following the BIDS App standard with a Graphical User Interface (GUI).

This means CMP3 can be executed in two different ways:

1. By running the BIDS App container image directly from the terminal or a script (See *Commandline Usage* section for more details).
2. By using its Graphical User Interface, designed to facilitate the configuration of all pipeline stages, the configuration of the BIDS App run and its execution, and the inspection of the different stage outputs with appropriate viewers (See *Graphical User Interface* section for more details) .

For more information about BIDS and BIDS-Apps, please consult the BIDS Website, the Online BIDS Specifications, and the BIDSApps Website. HeuDiConv can assist you in converting DICOM brain imaging data to BIDS. A nice tutorial can be found @ BIDS Tutorial Series: HeuDiConv Walkthrough .

5.2.1 Example BIDS dataset

For instance, a BIDS dataset with T1w, DWI and rs-fMRI images should adopt the following organization, naming, and file formats::

```
ds-example/

    README
    CHANGES
    participants.tsv
    dataset_description.json

    sub-01/
        anat/
            sub-01_T1w.nii.gz
            sub-01_T1w.json
        dwi/
            sub-01_dwi.nii.gz
            sub-01_dwi.json
            sub-01_dwi.bvec
            sub-01_dwi.bval
        func/
            sub-01_task-rest_bold.nii.gz
            sub-01_task-rest_bold.json

    ...

    sub-<subject_label>/
        anat/
            sub-<subject_label>_T1w.nii.gz
```

(continues on next page)

(continued from previous page)

```
            sub-<subject_label>_T1w.json
        ...
    ...
```

For an example of a dataset containing T1w, DWI and preprocessed EEG data, please check the public VEPCON dataset.

Important: Before using any BIDS App, we highly recommend you to validate your BIDS structured dataset with the free, online BIDS Validator.

5.3 Commandline Usage

`Connectome Mapper 3 (CMP3)` is distributed as a BIDS App which adopts the BIDS standard for data organization and takes as principal input the path of the dataset that is to be processed. The input dataset is required to be in valid `BIDS` format, and it must include at least a T1w or MPRAGE structural image and a DWI and/or resting-state fMRI image and/or preprocessed EEG data. See *Connectome Mapper 3 and the BIDS standard* page that provides links for more information about BIDS and BIDS-Apps as well as an example for dataset organization and naming.

Warning: As of `CMP3 v3.0.0-RC2`, the BIDS App includes a **tracking** system that anonymously reports the run of the BIDS App. This feature has been introduced to support us in the task of fund finding for the development of CMP3 in the future. However, users are still free to opt-out using the `--notrack` commandline argument.

Important: Since `v3.0.0-RC4`, configuration files adopt the `JSON` format. If you have your configuration files still in the *old* `INI` format, do not worry, the CMP3 BIDS App will convert them to the new `JSON` format automatically for you.

5.3.1 Commandline Arguments

The command to run `CMP3` follows the BIDS-Apps definition standard with additional options for loading pipeline configuration files.

Entrypoint script of the BIDS-App Connectome Mapper version v3.1.0

```
usage: connectomemapper3 [-h]
                         [--participant_label PARTICIPANT_LABEL [PARTICIPANT_LABEL ...]]
                         [--session_label SESSION_LABEL [SESSION_LABEL ...]]
                         [--anat_pipeline_config ANAT_PIPELINE_CONFIG]
                         [--dwi_pipeline_config DWI_PIPELINE_CONFIG]
                         [--func_pipeline_config FUNC_PIPELINE_CONFIG]
                         [--eeg_pipeline_config EEG_PIPELINE_CONFIG]
                         [--number_of_threads NUMBER_OF_THREADS]
                         [--number_of_participants_processed_in_parallel NUMBER_OF_
↪PARTICIPANTS_PROCESSED_IN_PARALLEL]
                         [--mrtrix_random_seed MRTRIX_RANDOM_SEED]
```

(continues on next page)

(continued from previous page)

```
                        [--ants_random_seed ANTS_RANDOM_SEED]
                        [--ants_number_of_threads ANTS_NUMBER_OF_THREADS]
                        [--fs_license FS_LICENSE] [--coverage] [--notrack]
                        [-v]
                        bids_dir output_dir {participant,group}
```

Positional Arguments

bids_dir The directory with the input dataset formatted according to the BIDS standard.

output_dir The directory where the output files should be stored. If you are running group level analysis this folder should be prepopulated with the results of the participant level analysis.

analysis_level Possible choices: participant, group

Level of the analysis that will be performed. Multiple participant level analyses can be run independently (in parallel) using the same output_dir.

Named Arguments

--participant_label The label(s) of the participant(s) that should be analyzed. The label corresponds to sub-<participant_label> from the BIDS spec (so it does not include "sub-"). If this parameter is not provided all subjects should be analyzed. Multiple participants can be specified with a space separated list.

--session_label **The label(s) of the session that should be analyzed.** The label corresponds to ses-<session_label> from the BIDS spec (so it does not include "ses-"). If this parameter is not provided all sessions should be analyzed. Multiple sessions can be specified with a space separated list.

--anat_pipeline_config Configuration .json file for processing stages of the anatomical MRI processing pipeline

--dwi_pipeline_config Configuration .json file for processing stages of the diffusion MRI processing pipeline

--func_pipeline_config Configuration .json file for processing stages of the fMRI processing pipeline

--eeg_pipeline_config Configuration .json file for processing stages of the eeg processing pipeline

--number_of_threads The number of OpenMP threads used for multi-threading by Freesurfer (Set to [Number of available CPUs -1] by default).

--number_of_participants_processed_in_parallel The number of subjects to be processed in parallel (One by default).

Default: 1

--mrtrix_random_seed Fix MRtrix3 random number generator seed to the specified value

--ants_random_seed Fix ANTS random number generator seed to the specified value

--ants_number_of_threads Fix number of threads in ANTs. If not specified ANTs will use the same number as the number of OpenMP threads (see —-*number_of_threads* option flag)

--fs_license Freesurfer license.txt

--coverage Run connectomemapper3 with coverage

Default: False

--notrack Do not send event to Google analytics to report BIDS App execution, which is enabled by default.

Default: False

-v, --version show program's version number and exit

Important: Before using any BIDS App, we highly recommend you to validate your BIDS structured dataset with the free, online BIDS Validator.

5.3.2 Participant Level Analysis

You can run `CMP3` using the lightweight Docker or Singularity wrappers we created for convenience or you can interact directly with the Docker / Singularity Engine via the docker or singularity run command.

New in v3.0.2

You can now be aware about the adverse impact of your processing on the environment !

With the new *–track_carbon_footprint* option of the *connectomemapper3_docker* and *connectomemapper3_singularity* BIDS App python wrappers, you can use codecarbon to estimate the amount of carbon dioxide (CO2) produced to execute the code by the computing resources and save the results in `<bids_dir>/code/emissions.csv`.

Then, to visualize, interpret and track the evolution of the CO2 emissions incurred, you can use the visualization tool of *codecarbon* aka *carbonboard* that takes as input the *.csv* created:

```
$ carbonboard --filepath="<bids_dir>/code/emissions.csv" --port=xxxx
```

With the wrappers

When you run `connectomemapper3_docker`, it will generate a Docker command line for you, print it out for reporting purposes, and then execute it without further action needed, e.g.:

```
$ connectomemapper_docker \
    "/home/user/data/ds001" "/home/user/data/ds001/derivatives" \
    participant --participant_label 01 --session_label 01 \
    --fs_license "/usr/local/freesurfer/license.txt" \
    --config_dir "/home/user/data/ds001/code" \
    --track_carbon_footprint \
    --anat_pipeline_config "ref_anatomical_config.json" \
    (--dwi_pipeline_config "ref_diffusion_config.json" \)
    (--func_pipeline_config "ref_fMRI_config.json" \)
    (--eeg_pipeline_config "ref_EEG_config.json" \)
    (--number_of_participants_processed_in_parallel 1)
```

When you run `connectomemapper3_singularity`, it will generate a Singularity command line for you, print it out for reporting purposes, and then execute it without further action needed, e.g.:

```
$ connectomemapper3_singularity \
    "/home/user/data/ds001" "/home/user/data/ds001/derivatives" \
    participant --participant_label 01 --session_label 01 \
    --fs_license "/usr/local/freesurfer/license.txt" \
    --config_dir "/home/user/data/ds001/code" \
    --track_carbon_footprint \
    --anat_pipeline_config "ref_anatomical_config.json" \
    (--dwi_pipeline_config "ref_diffusion_config.json" \)
    (--func_pipeline_config "ref_fMRI_config.json" \)
    (--eeg_pipeline_config "ref_EEG_config.json" \)
    (--number_of_participants_processed_in_parallel 1)
```

With the Docker / Singularity Engine

If you need a finer control over the container execution, or you feel comfortable with the Docker or Singularity Engine, avoiding the extra software layer of the wrapper might be a good decision.

Docker

For instance, the previous call to the `connectomemapper3_docker` wrapper corresponds to:

```
$ docker run -t --rm -u $(id -u):$(id -g) \
        -v /home/user/data/ds001:/bids_dir \
        -v /home/user/data/ds001/derivatives:/output_dir \
        (-v /usr/local/freesurfer/license.txt:/bids_dir/code/license.txt) \
        sebastientourbier/connectomemapper-bidsapp:v3.1.0 \
        /bids_dir /output_dir participant --participant_label 01 (--session_label 01) \
        --anat_pipeline_config /bids_dir/code/ref_anatomical_config.json \
        (--dwi_pipeline_config /bids_dir/code/ref_diffusion_config.json \)
        (--func_pipeline_config /bids_dir/code/ref_fMRI_config.json \)
        (--eeg_pipeline_config /bids_dir/code/ref_EEG_config.json \)
        (--number_of_participants_processed_in_parallel 1)
```

Singularity

For instance, the previous call to the `connectomemapper3_singularity` wrapper corresponds to:

```
$ singularity run  --containall \
        --bind /home/user/data/ds001:/bids_dir \
        --bind /home/user/data/ds001/derivatives:/output_dir \
        --bind /usr/local/freesurfer/license.txt:/bids_dir/code/license.txt \
        library://connectomicslab/default/connectomemapper-bidsapp:v3.1.0 \
        /bids_dir /output_dir participant --participant_label 01 (--session_label 01) \
        --anat_pipeline_config /bids_dir/code/ref_anatomical_config.json \
        (--dwi_pipeline_config /bids_dir/code/ref_diffusion_config.json \)
        (--func_pipeline_config /bids_dir/code/ref_fMRI_config.json \)
        (--eeg_pipeline_config /bids_dir/code/ref_EEG_config.json \)
        (--number_of_participants_processed_in_parallel 1)
```

Note: The local directory of the input BIDS dataset (here: /home/user/data/ds001) and the output directory (here: /home/user/data/ds001/derivatives) used to process have to be mapped to the folders /bids_dir and /output_dir respectively using the docker -v / singularity --bind run option.

Important: The user is requested to use its own Freesurfer license (available here). CMP expects by default to find a copy of the FreeSurfer license.txt in the code/ folder of the BIDS directory. However, one can also mount a freesurfer license.txt with the docker -v / singularity --bind run option. This file can be located anywhere on the computer (as in the example above, i.e. /usr/local/freesurfer/license.txt) to the code/ folder of the BIDS directory inside the docker container (i.e. /bids_dir/code/license.txt).

Note: At least a configuration file describing the processing stages of the anatomical pipeline should be provided. Diffusion and/or Functional MRI pipeline are performed only if a configuration file is set. The generation of such configuration files, the execution of the BIDS App docker image and output inpection are facilitated through the use of the Connectome Mapper GUI, i.e. cmpbidsappmanager (see dedicated documentation page)

5.3.3 Debugging

Logs are saved into <output dir>/cmp/sub-<participant_label>/sub-<participant_label>_log.txt.

5.3.4 Already have Freesurfer outputs?

If you have already Freesurfer v5 / v6 output data available, CMP3 can use them if there are properly placed in your output / derivatives directory. Since v3.0.3, CMP3 expects to find a freesurfer-7.1.1, so make sure that your derivatives are organized as follows:

```
your_bids_dataset
  |______ derivatives/
  |         |______ freesurfer-7.1.1/
  |                   |______ sub-01[_ses-01]/
  |                   |           |______ label/
  |                   |           |______ mri/
  |                   |           |______ surf/
  |                   |           |______ ...
  |                   |______ ...
  |______ sub-01/
  |______ ...
```

5.3.5 Support, bugs and new feature requests

If you need any support or have any questions, you can post to the CMTK-users group.

All bugs, concerns and enhancement requests for this software are managed on GitHub and can be submitted at https://github.com/connectomicslab/connectomemapper3/issues.

5.3.6 Not running on a local machine?

If you intend to run `CMP3` on a remote system such as a high-performance computing cluster where Docker is not available due to root privileges, a Singularity image is also built for your convenience and available on Sylabs.io. Please see instructions at *Running on a cluster (HPC)*.

Also, you will need to make your data available within that system first. Comprehensive solutions such as Datalad will handle data transfers with the appropriate settings and commands. Datalad also performs version control over your data. A tutorial is provided in *Adopting Datalad for collaboration*.

5.4 Graphical User Interface

5.4.1 Introduction

Connectome Mapper 3 comes with a Graphical User Interface, the Connectome Mapper BIDS App manager, designed to facilitate the configuration of all pipeline stages, the configuration of the BIDS App run and its execution, and the inspection of the different stage outputs with appropriate viewers.

5.4.2 Start the Graphical User Interface

In a terminal, enter to following:

```
$ source activate py39cmp-gui
```

or:

```
$ conda activate py39cmp-gui
```

Please see Section *Installation* for more details about installation.

After activation of the conda environment, start the graphical user interface called `Connectome Mapper 3` BIDS App Manager

```
$ cmpbidsappmanager
```

Note: The main window would be blank until you select the BIDS dataset.

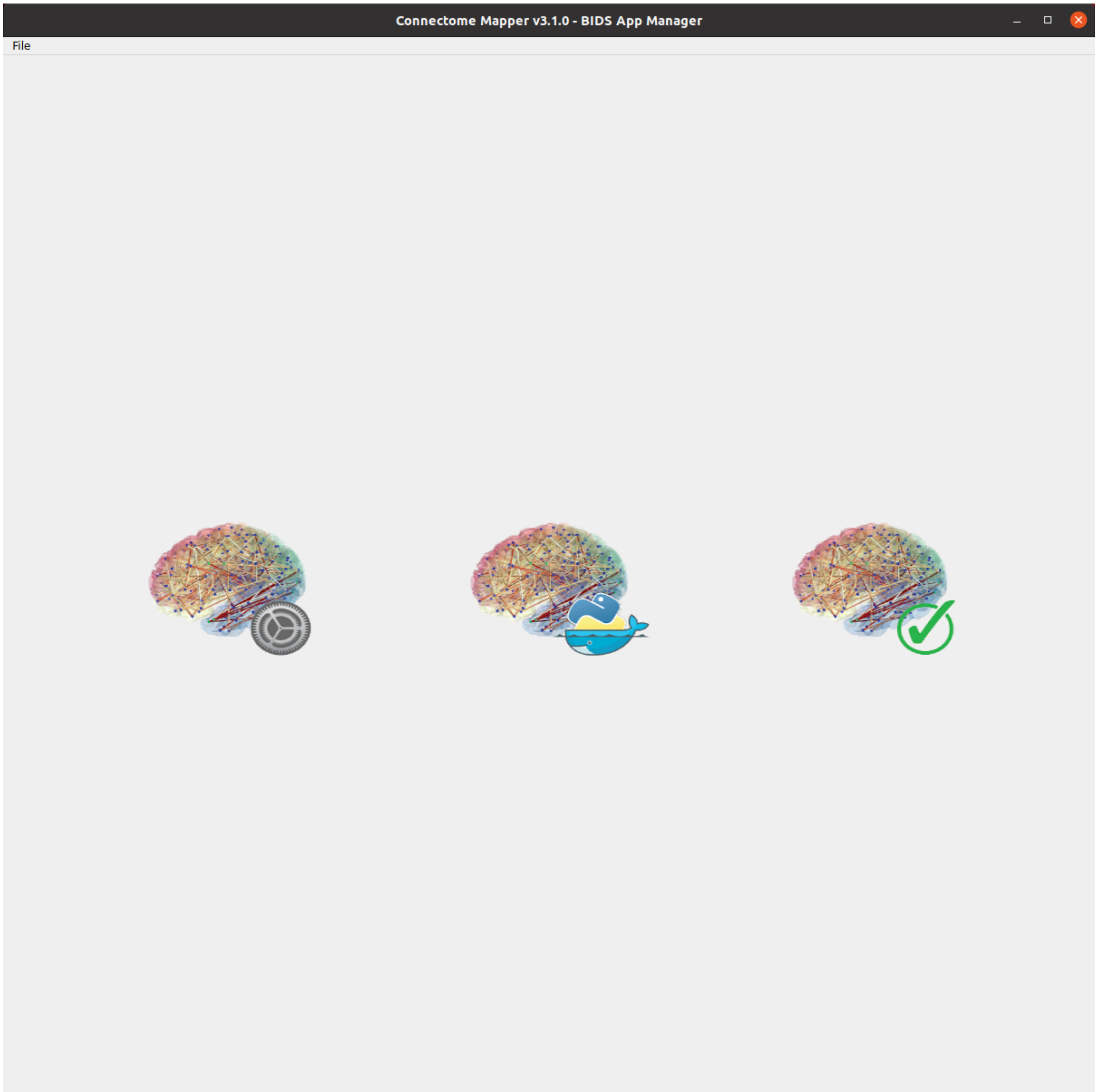


Fig. 1: Main window of the Connectome Mapper BIDS App Manager

5.4.3 Load a BIDS dataset

- Click on `File -> Load BIDS dataset...` in the menu bar of the main window. Note that on Mac, Qt turns this menu bar into the native menu bar (top of the screen).

  The `Connectome Mapper 3` BIDS App Manager gives you two different options:

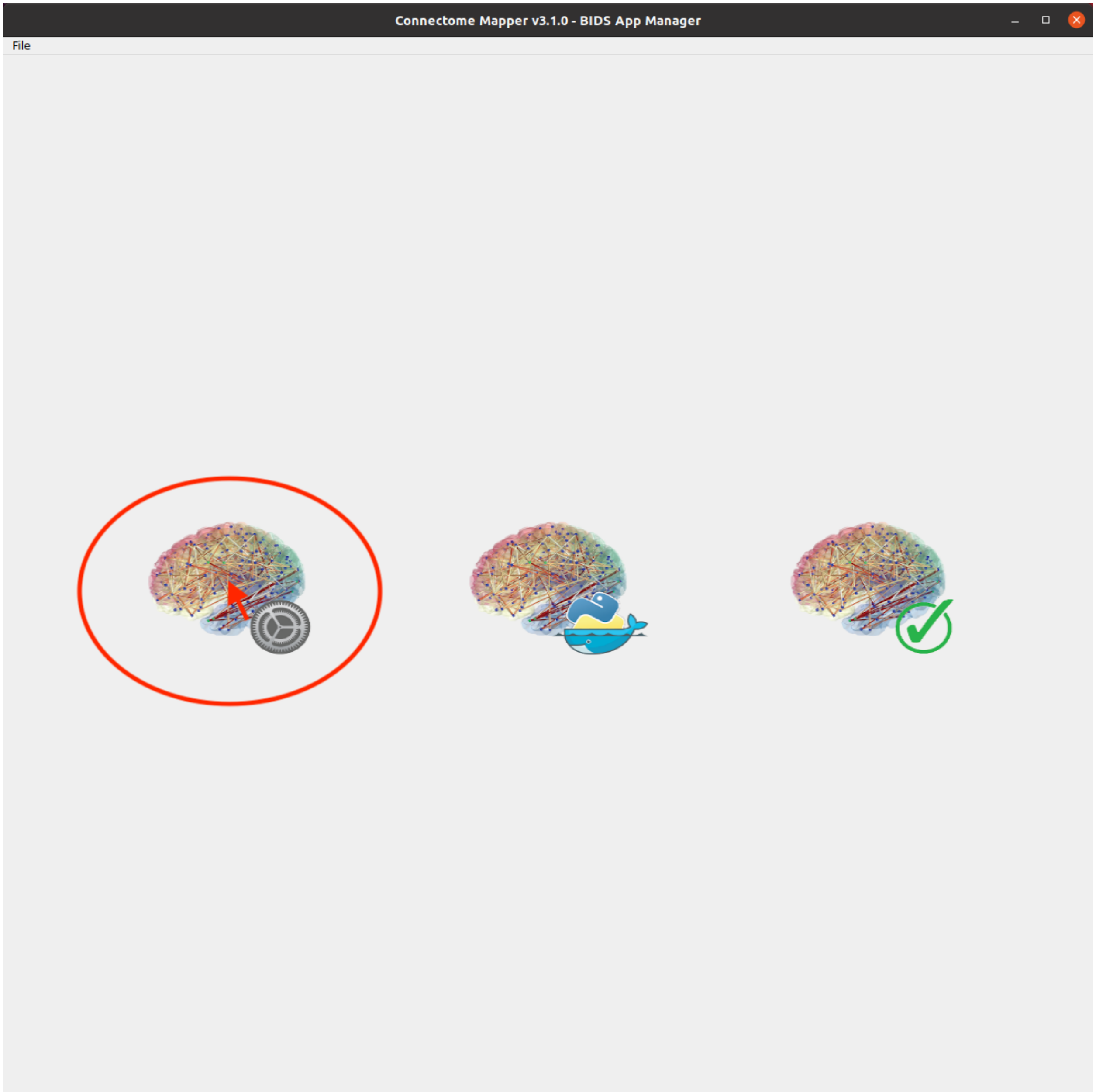
  - `Load BIDS dataset`: load a BIDS dataset stored locally. You only have to select the root directory of your valid BIDS dataset (see note below)
  - `Install Datalad BIDS dataset`: create a new datalad/BIDS dataset locally from an existing local or remote datalad/BIDS dataset (This is a feature under development) If ssh connection is used, make sure to enable the "install via ssh" and to provide all connection details (IP address / Remote host name, remote user, remote password)

Note: The input dataset MUST be a valid BIDS structured dataset and must include at least one T1w or MPRAGE structural image. We highly recommend that you validate your dataset with the free, online BIDS Validator.

5.4.4 Pipeline stage configuration

Start the Configurator Window

- From the main window, click on the left button to start the Configurator Window.



- The window of the Connectome Mapper BIDS App Configurator will appear, which will assist you note only in configuring the pipeline stages (each pipeline has a tab panel), but also in creating appropriate configuration files which could be used outside the Graphical User Interface.

The outputs depend on the chosen parameters.

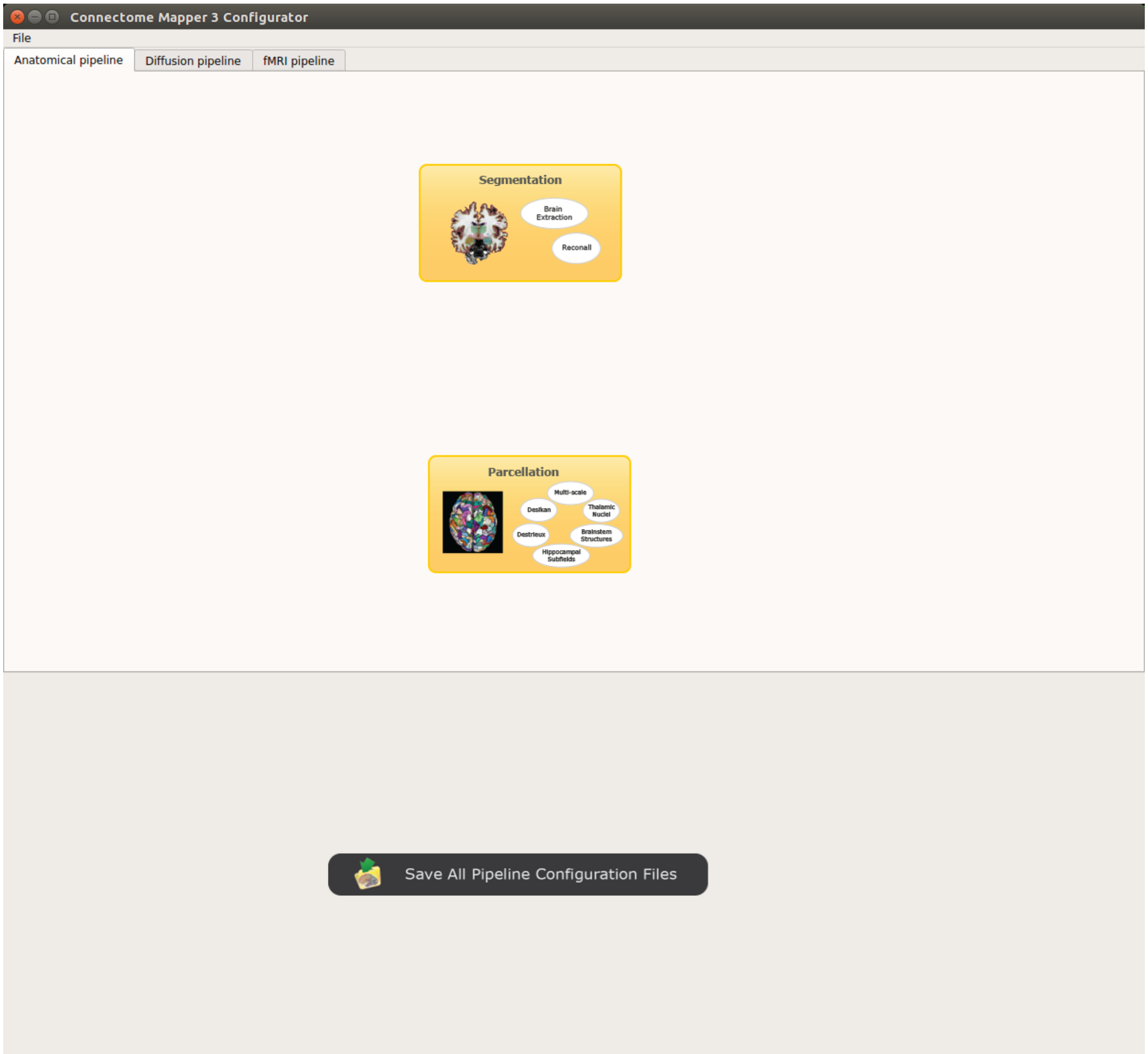


Fig. 2: Configurator Window of the Connectome Mapper

Anatomical pipeline stages

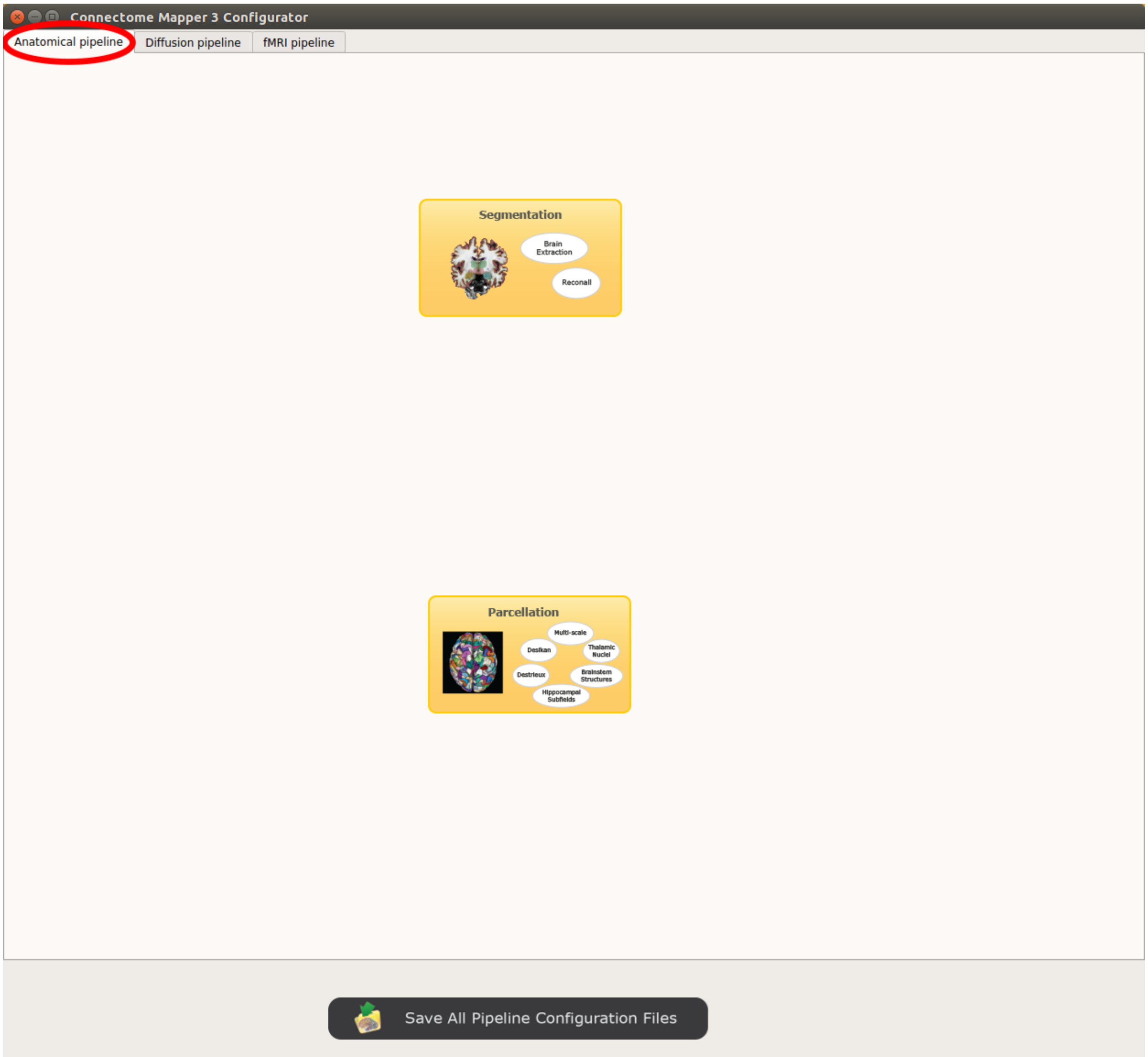
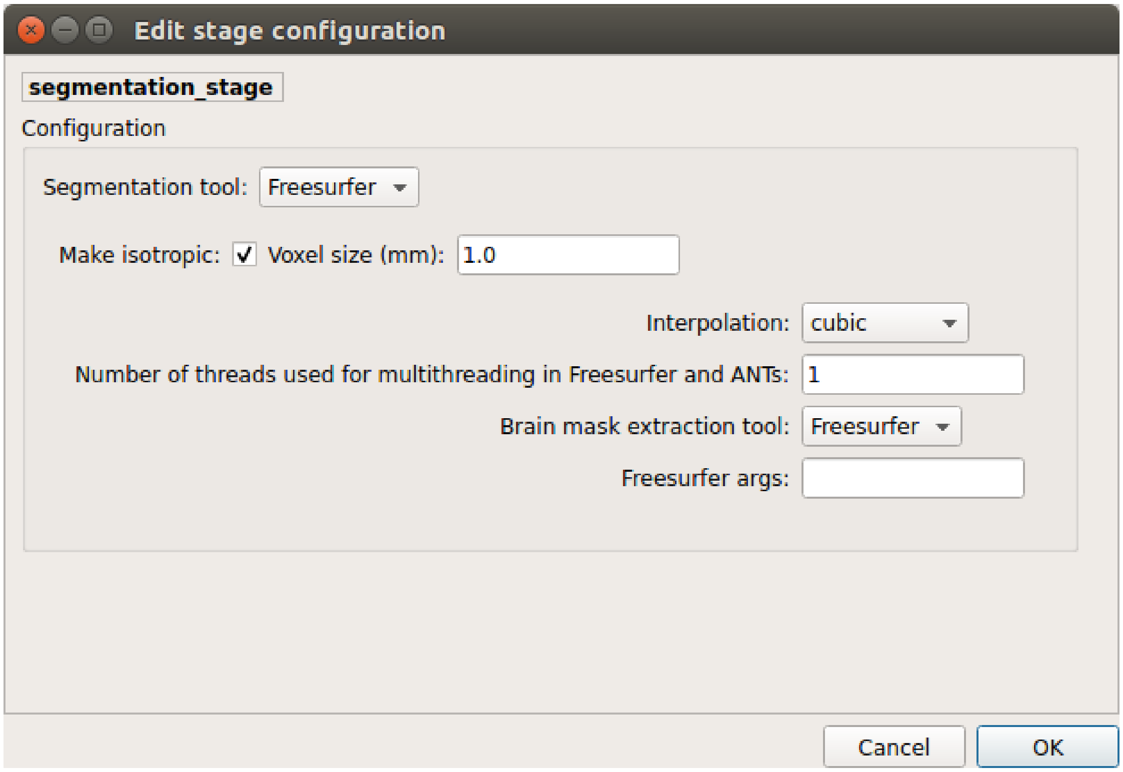


Fig. 3: Panel for configuration of anatomical pipeline stages

Segmentation

Prior to Lausanne parcellation, CMP3 relies on **Freesurfer** for the segmentation of the different brain tissues and the reconstruction of the cortical surfaces. If you plan to use a custom parcellation, you will be required here to specify the pattern of the different existing segmentation files that follows BIDS derivatives (See *Custom segmentation*).

Freesurfer

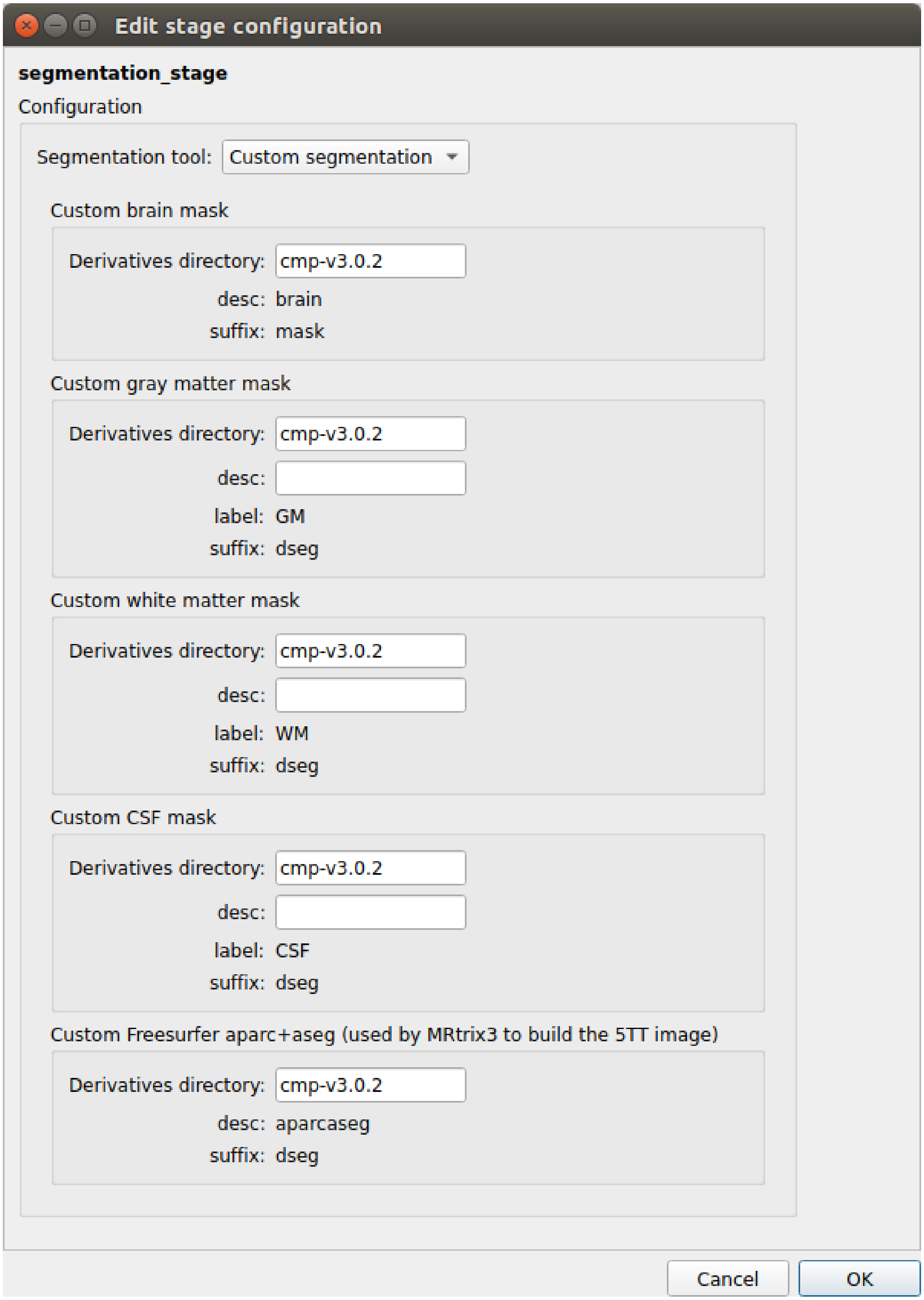


- *Number of threads:* used to specify how many threads are used for parallelization
- *Brain extraction tools:* alternative brain extraction methods injected in Freesurfer
- *Freesurfer args:* used to specify extra Freesurfer processing options

Note: If you have already Freesurfer v5 / v6 / v7 output data available, CMP3 can use them if there are placed in your output / derivatives directory. Note however that since `v3.0.3`, CMP3 expects to find a `freesurfer-7.1.1`, so make sure that your derivatives are organized as follows:

```
your_bids_dataset
  derivatives/
    freesurfer-7.1.1/
      sub-01[_ses-01]/
        label/
        mri/
        surf/
        ...
      ...
  sub-01/
  ...
```

Custom segmentation



You can use any parcellation scheme of your choice as long as you provide a list of segmentation files organized following the BIDS derivatives specifications for segmentation files, provide appropriate `.tsv` sidecar files that describes the index/label/color mapping of the parcellation, and adopt the `atlas-<label>` entity to encode the name of the atlas, i.e:

```
<derivatives_directory>/
  sub-<participant_label>/
    anat/
      <source_entities>_desc-brain_mask.nii.gz
      <source_entities>_label-GM[_desc-<label>]_dseg.nii.gz
      <source_entities>_label-WM[_desc-<label>]_dseg.nii.gz
      <source_entities>_label-CSF[_desc-<label>]_dseg.nii.gz
      <source_entities>_desc-aparcaseg_dseg.nii.gz
```

The `desc` BIDS entity can be used to target specific mask and segmentation files.

For instance, the configuration above would allows us to re-use the outputs of the anatomical pipeline obtained with the previous `v3.0.2` version of CMP3:

```
your_bids_dataset
  derivatives/
    cmp-v3.0.2/
      sub-01/
        anat/
          sub-01_desc-brain_mask.nii.gz
          sub-01_label-GM_dseg.nii.gz
          sub-01_label-WM_dseg.nii.gz
          sub-01_label-CSF_dseg.nii.gz
          sub-01_desc-aparcaseg_dseg.nii.gz
          ...
      ...
  sub-01/
  ...
```

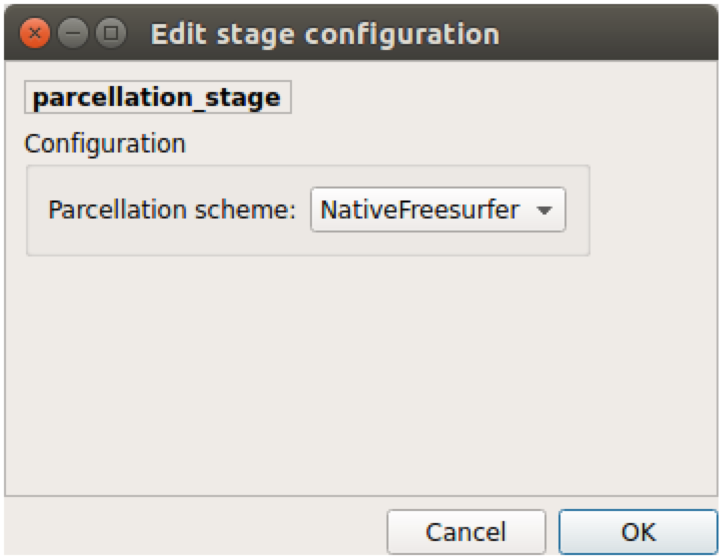
Important: If you plan to use either Anatomically Constrained or Particle Filtering tractography, you will still require to have Freesurfer 7 output data available in your output / derivatives directory, as described the above note in *Freesurfer*.

Parcellation

Generates the Native Freesurfer or Lausanne2018 parcellation from Freesurfer data. Alternatively, since `v3.0.3` you can use your own custom parcellation files.

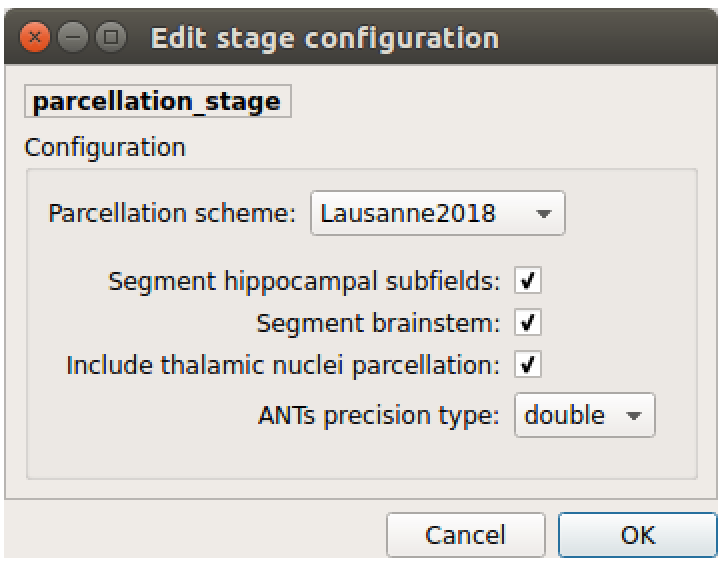
Parcellation scheme

- *NativeFreesurfer:*



Atlas composed of 83 regions from the Freesurfer aparc+aseg file

- *Lausanne2018:*



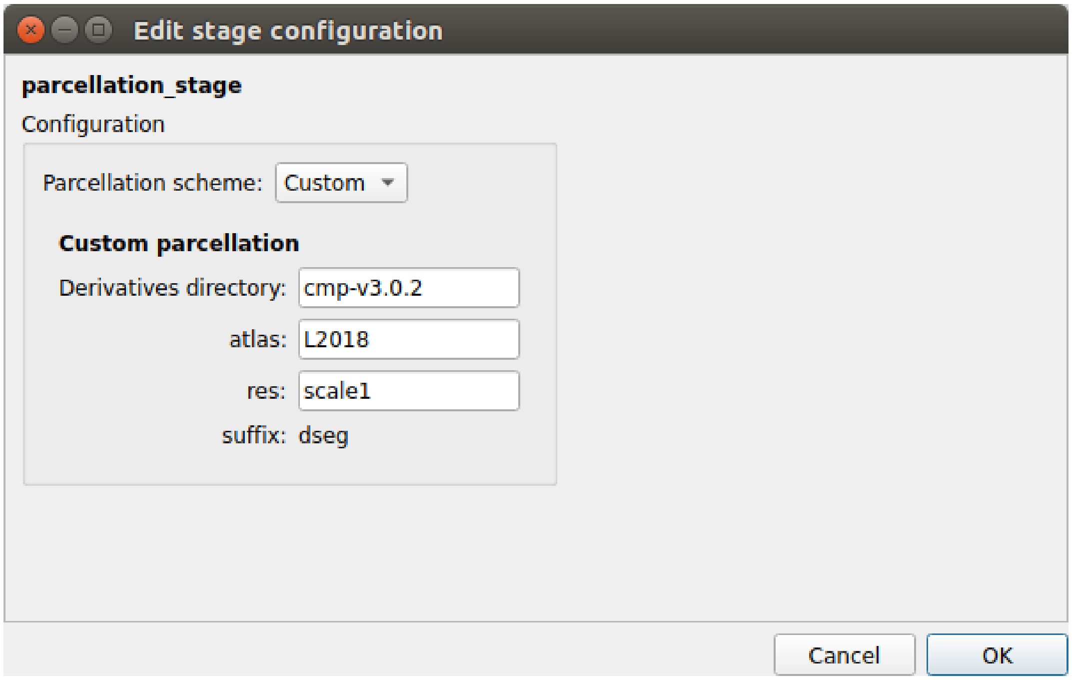
New version of Lausanne parcellation atlas, corrected, and extended with 7 thalamic nuclei, 12 hippocampal subfields, and 4 brainstem sub-structure per hemisphere

---

Since v3.0.0, Lausanne2018 parcellation has completely replaced the old Lausanne2008 parcellation.

As it provides improvements in the way Lausanne parcellation label are generated, any code and data related to Lausanne2008 has been removed. If you still wish to use this old parcellation scheme, please use `v3.0.0-RC4` which is the last version that supports it.

- *Custom:*



You can use any parcellation scheme of your choice as long as they follow the BIDS derivatives specifications for segmentation files, provide appropriate `.tsv` sidecar files that describes the index/label/color mapping of the parcellation, and adopt the `atlas-<label>` entity to encode the name of the atlas, i.e:

```
<derivatives_directory>/
  sub-<participant_label>/
    anat/
      <source_entities>[_space-<space>]_atlas-<label>[_res-<label>]_dseg.
↪nii.gz
      <source_entities>[_space-<space>]_atlas-<label>[_res-<label>]_dseg.tsv
```

The `res` BIDS entity allows the differentiation between multiple scales of the same atlas.

For instance, the above configuration would allows us to re-use the scale 1 of the Lausanne parcellation generated by the anatomical pipeline obtained of the previous `v3.0.2` version of CMP3:

```
your_bids_dataset
  derivatives/
    cmp-v3.0.2/
      sub-01/
        anat/
          sub-01_atlas-L2018_res-scale1_dseg.nii.gz
          sub-01_atlas-L2018_res-scale1_dseg.tsv
          ...
      ...
  sub-01/
  ...
```

Diffusion pipeline stages

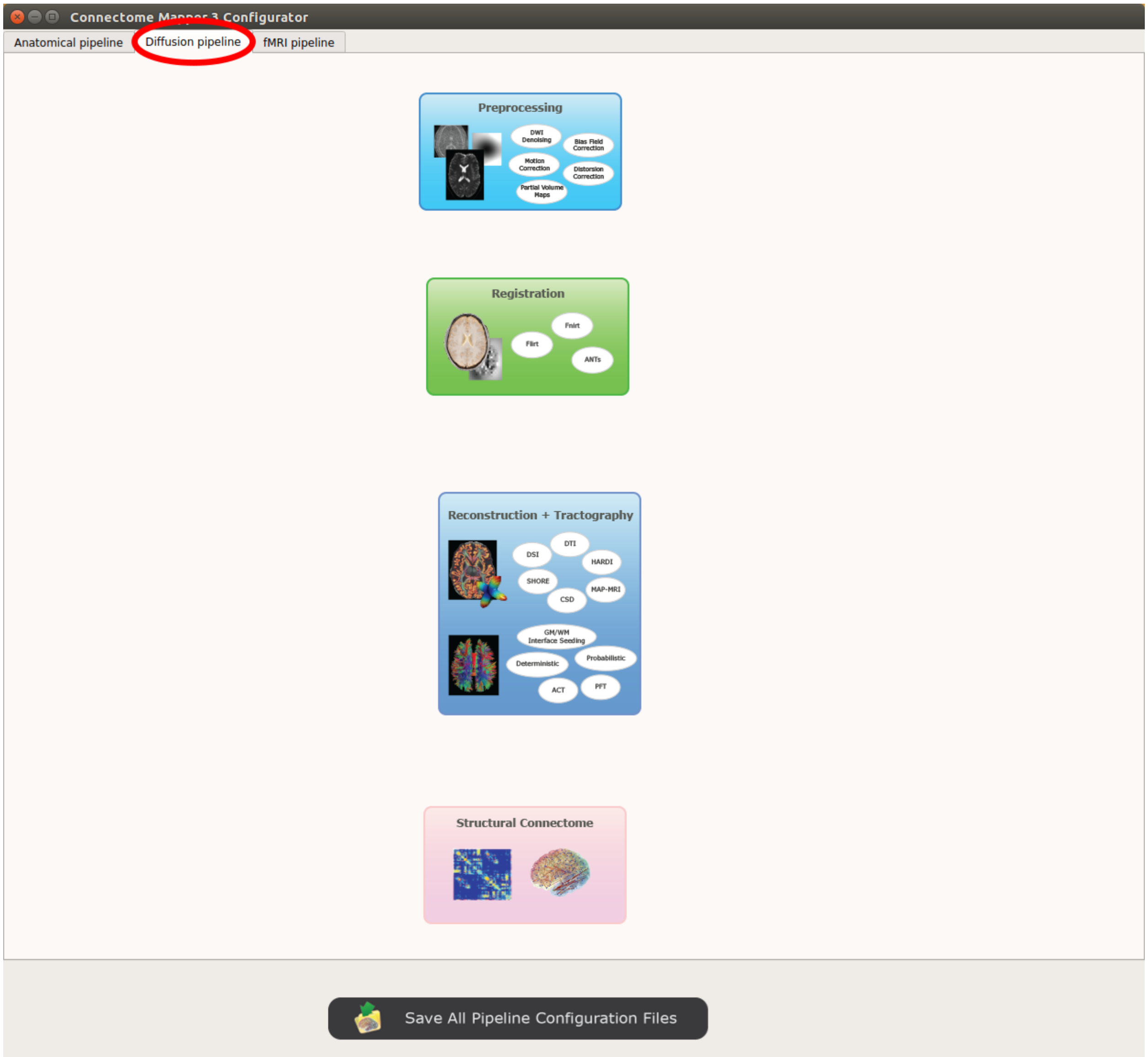
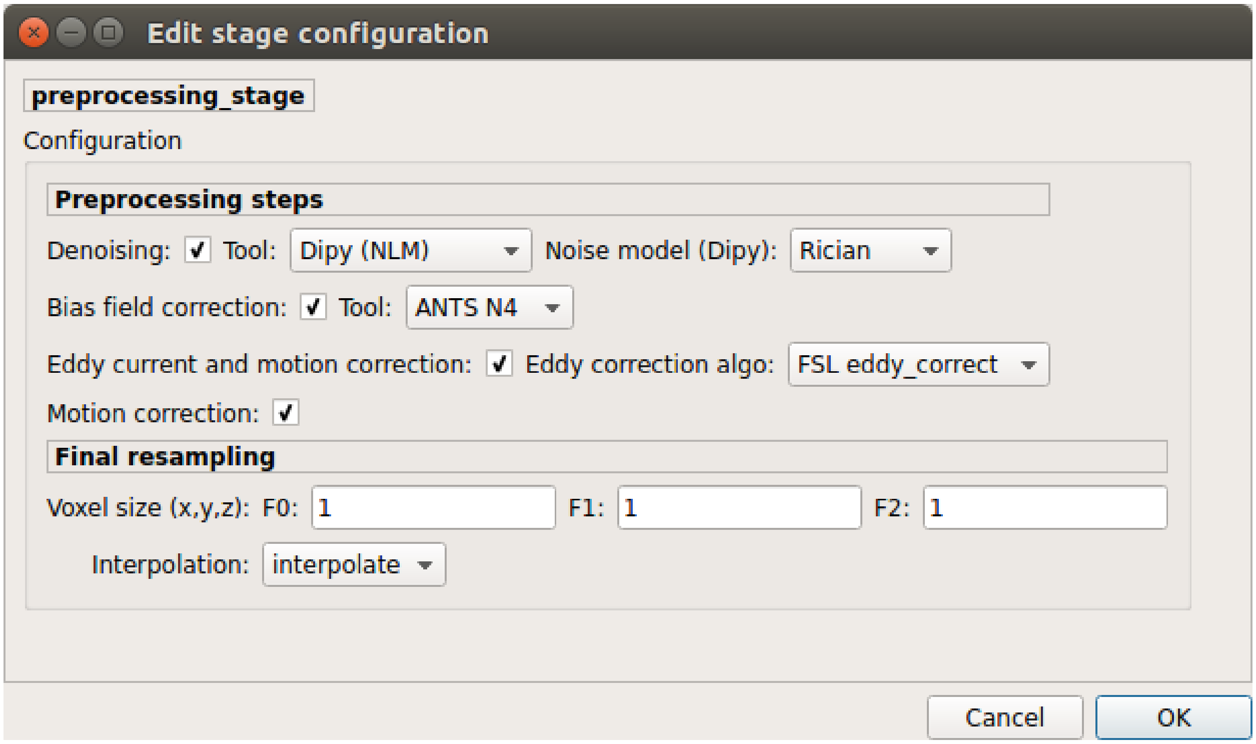


Fig. 4: Panel for configuration of diffusion pipeline stages

Preprocessing

Preprocessing includes denoising, bias field correction, motion and eddy current correction for diffusion data.



Denoising

Remove noise from diffusion images using (1) MRtrix3 MP-PCA method or (2) Dipy Non-Local Mean (NLM) denoising with Gaussian or Rician noise models

Bias field correction

Remove intensity inhomogeneities due to the magnetic resonance bias field using (1) MRtrix3 N4 bias field correction or (2) the bias field correction provided by FSL FAST

Motion correction

Aligns diffusion volumes to the b0 volume using FSL's MCFLIRT

Eddy current correction

Corrects for eddy current distortions using FSL's Eddy correct tool

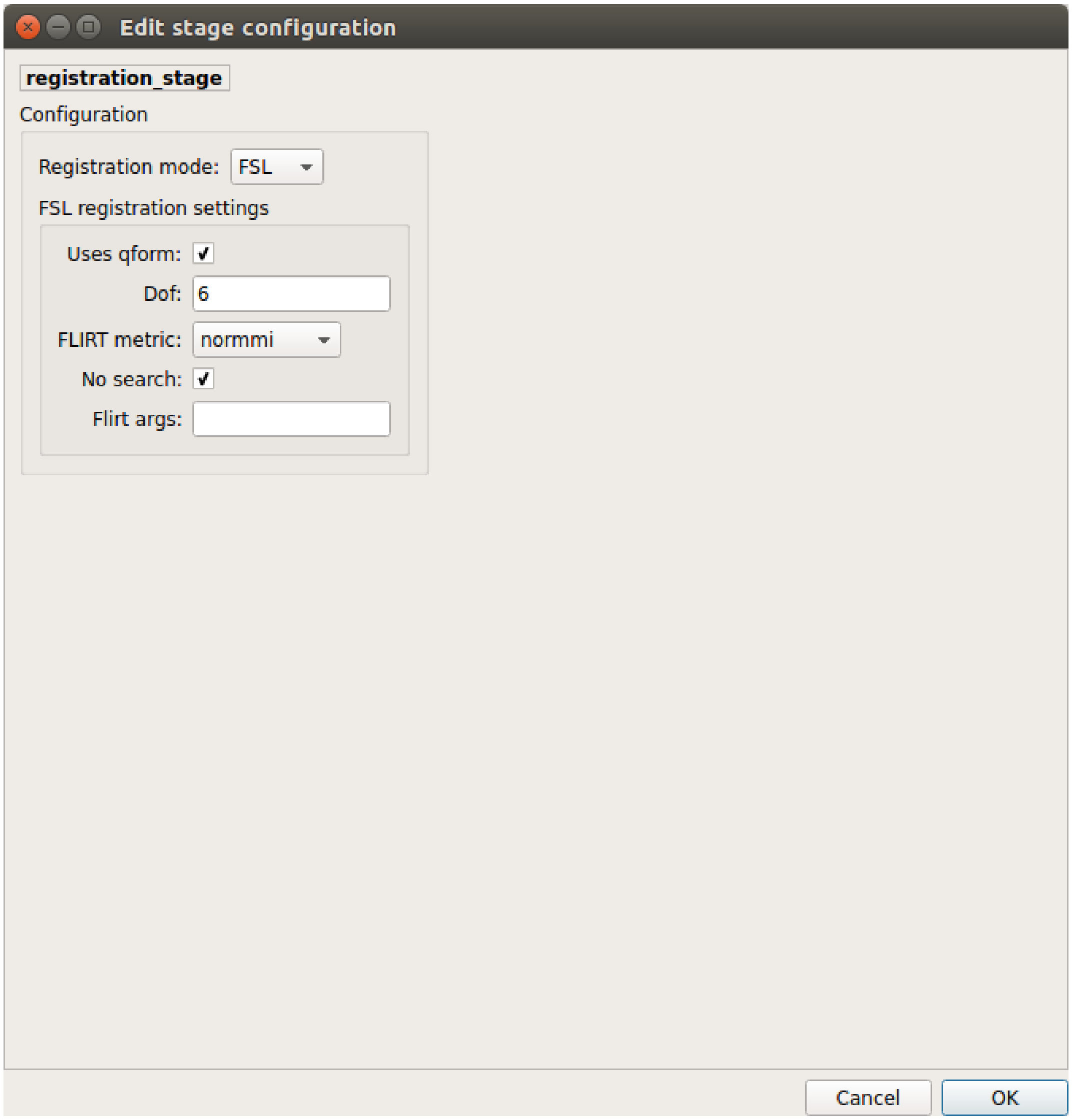
Resampling

Resample morphological and diffusion data to F0 x F1 x F2 mm^3

Registration

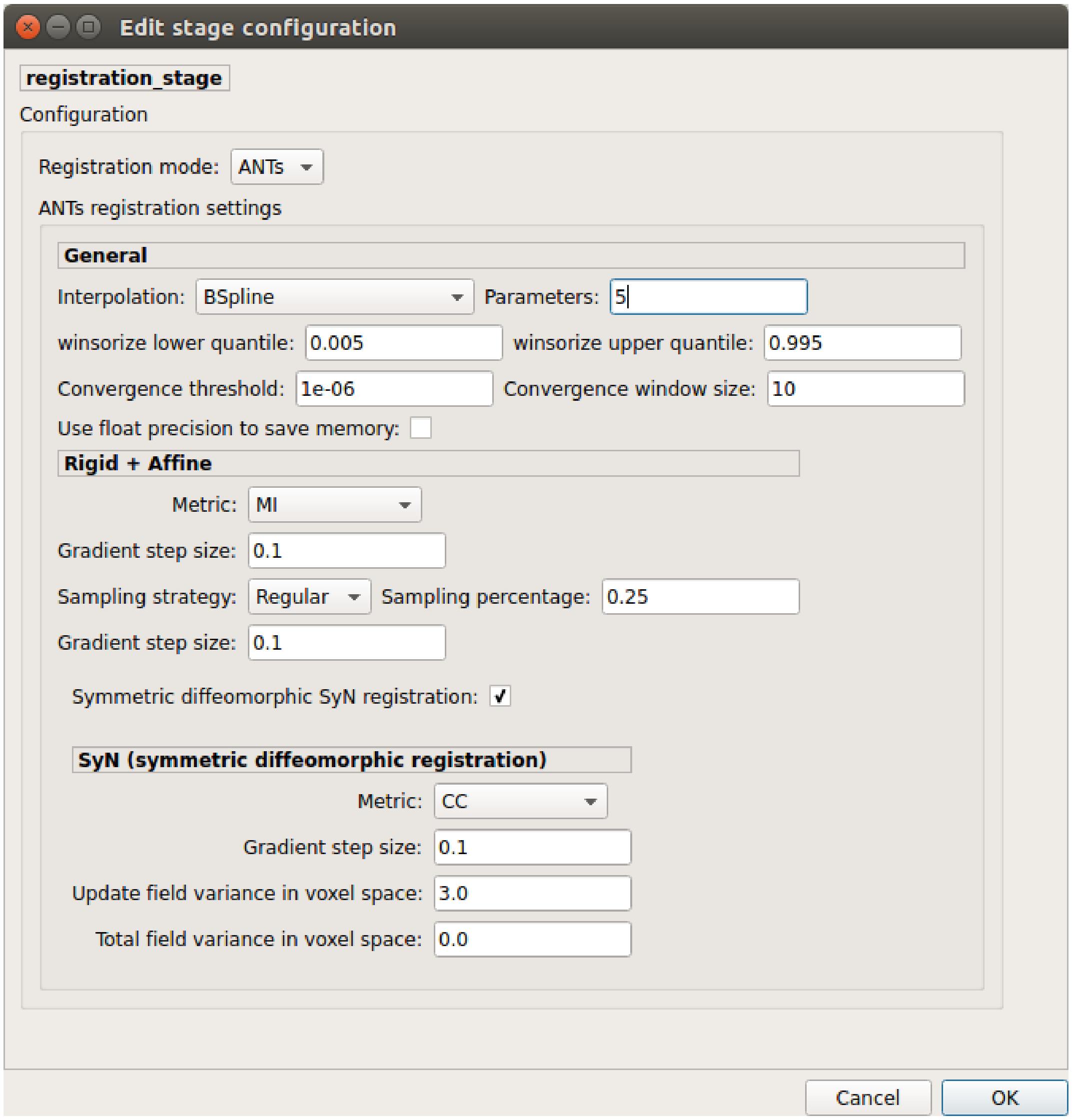
Registration mode

- FSL (Linear):



Perform linear registration from T1 to diffusion b0 using FSL's flirt

- Non-linear (ANTS):



Perform symmetric diffeomorphic SyN registration from T1 to diffusion b=0

Diffusion reconstruction and tractography

Perform diffusion reconstruction and local deterministic or probabilistic tractography based on several tools. ROI dilation is required to map brain connections when the tracking only operates in the white matter.

Reconstruction tool

Dipy: perform SHORE, tensor, CSD and MAP-MRI reconstruction

- SHORE:

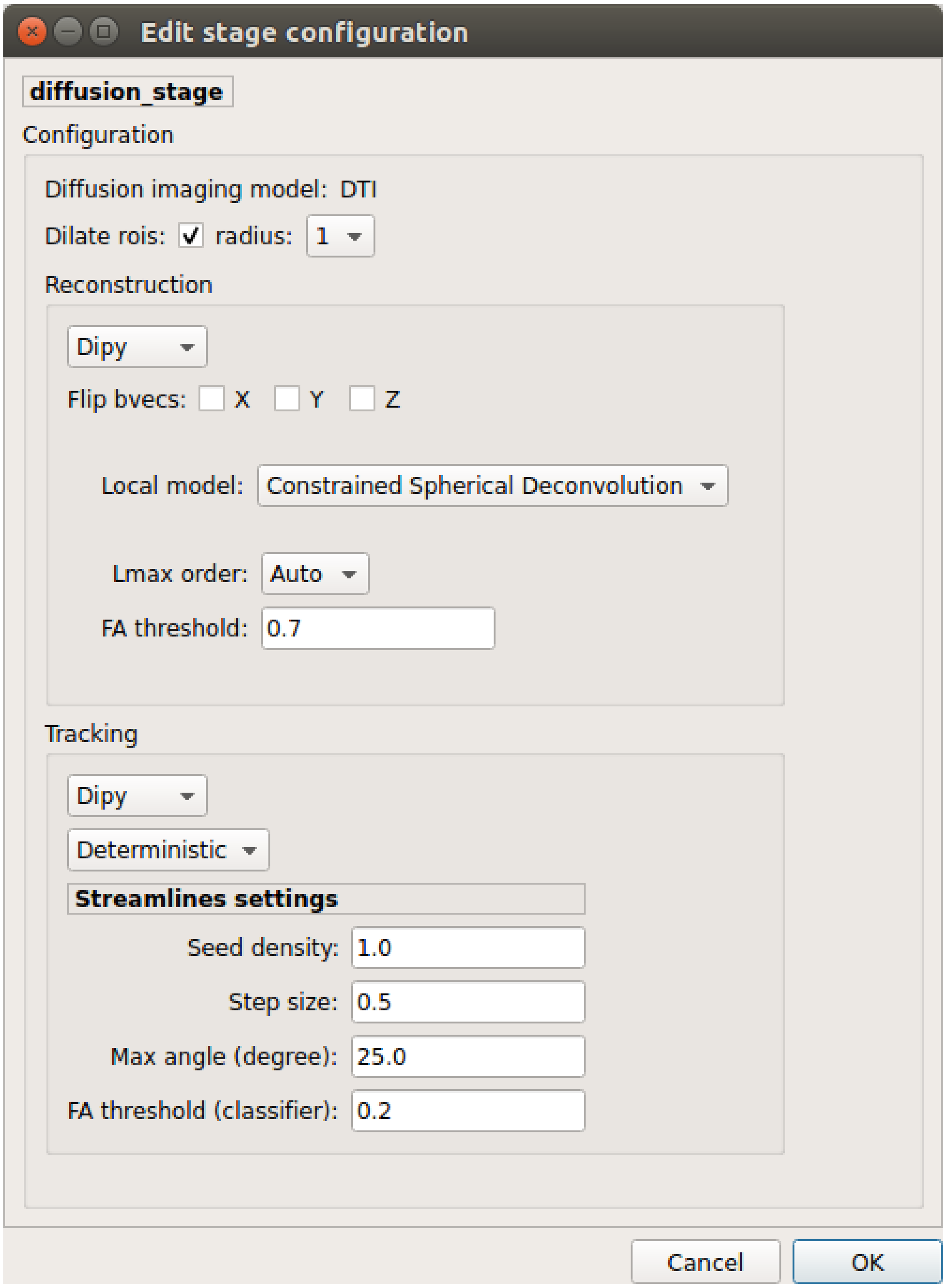
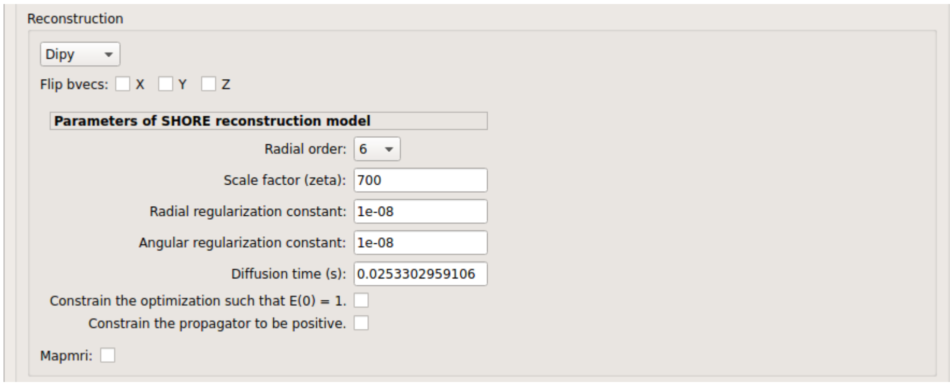
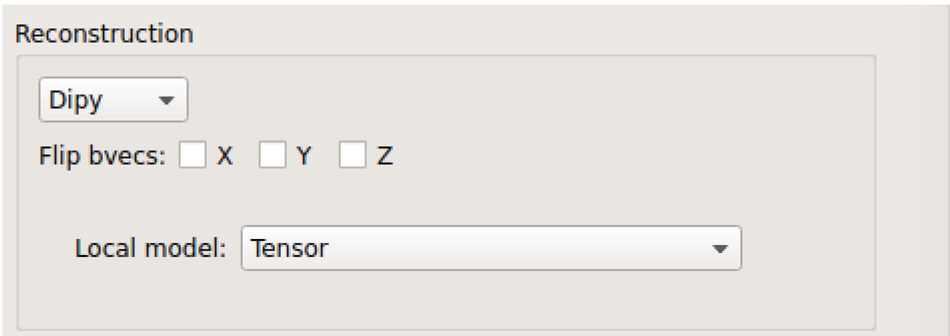


Fig. 5: Diffusion stage configuration window



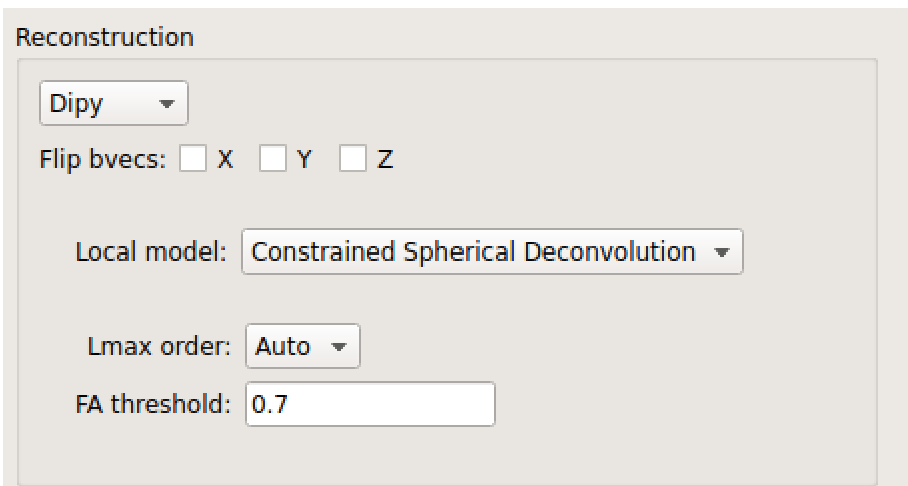
SHORE performed only on DSI data

- Tensor:



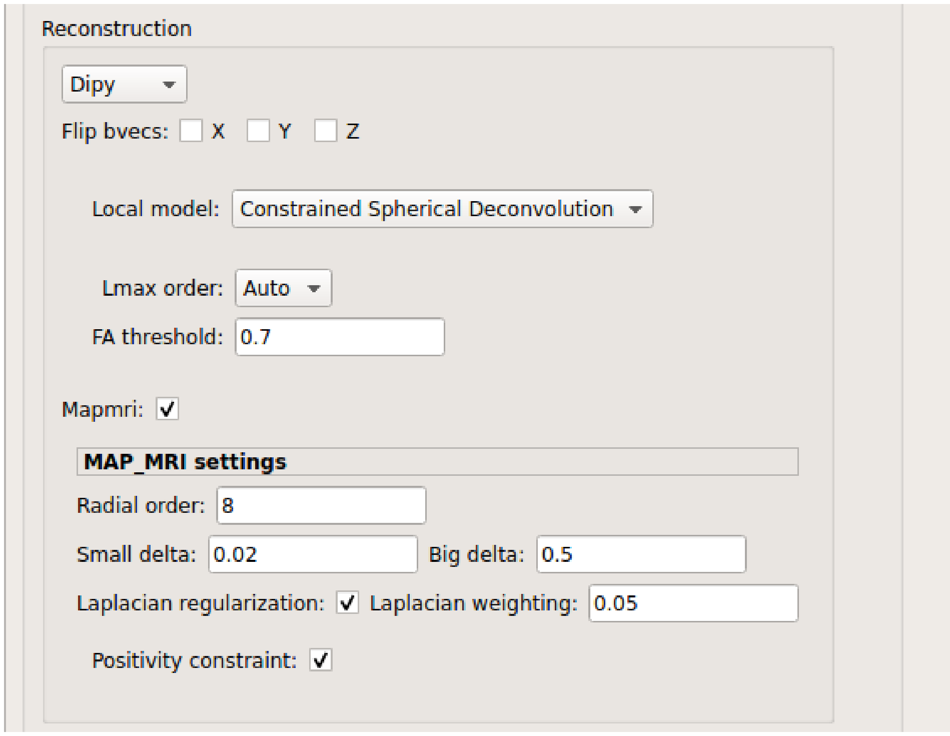
Tensor performed only on DTI data

- CSD:



CSD performed on DTI and multi-shell data

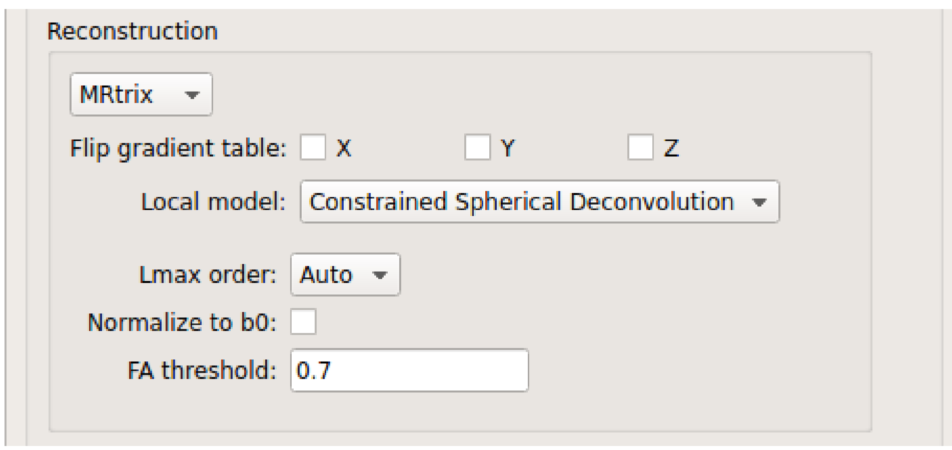
- MAP_MRI:



MAP-MRI performed only on multi-shell data

MRtrix: perform CSD reconstruction.

- CSD:

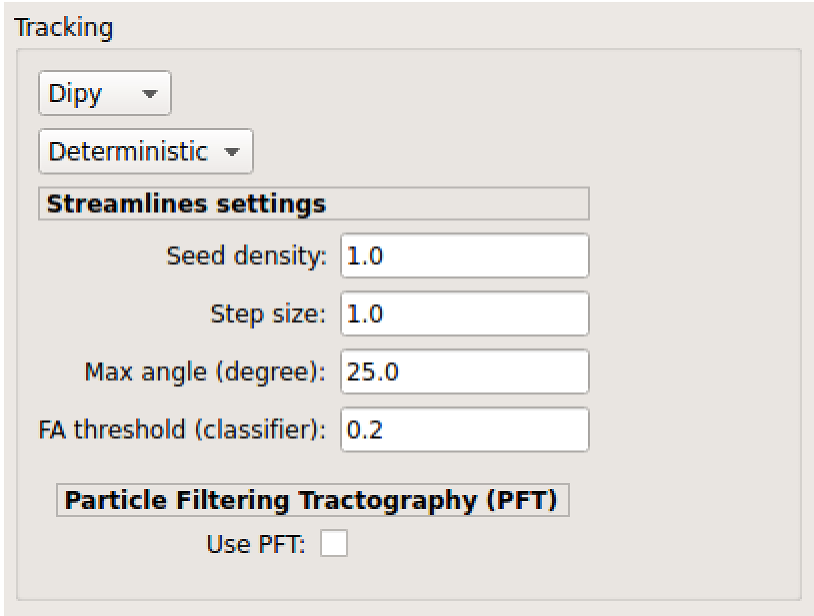


CSD performed on DTI and multi-shell data

Tractography tool

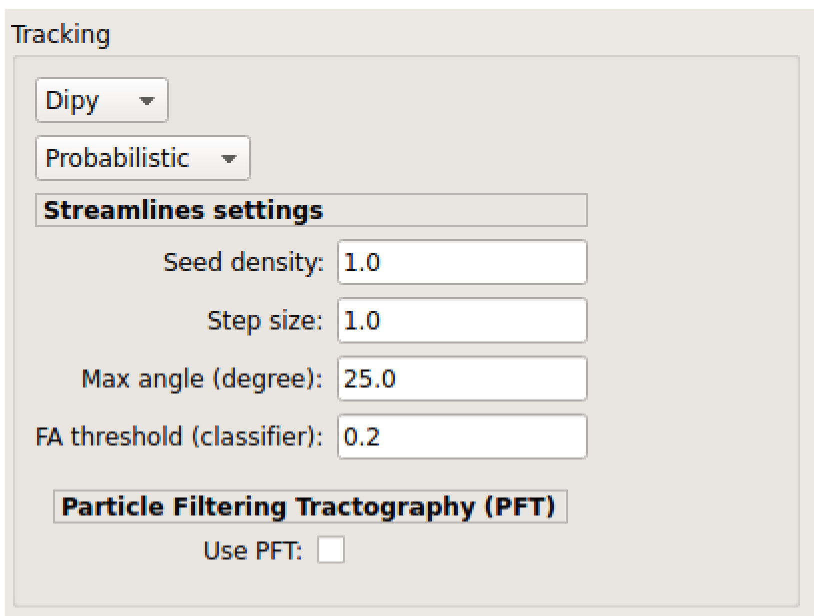
Dipy: perform deterministic and probabilistic fiber tracking as well as particle filtering tractography.

- Deterministic tractography:



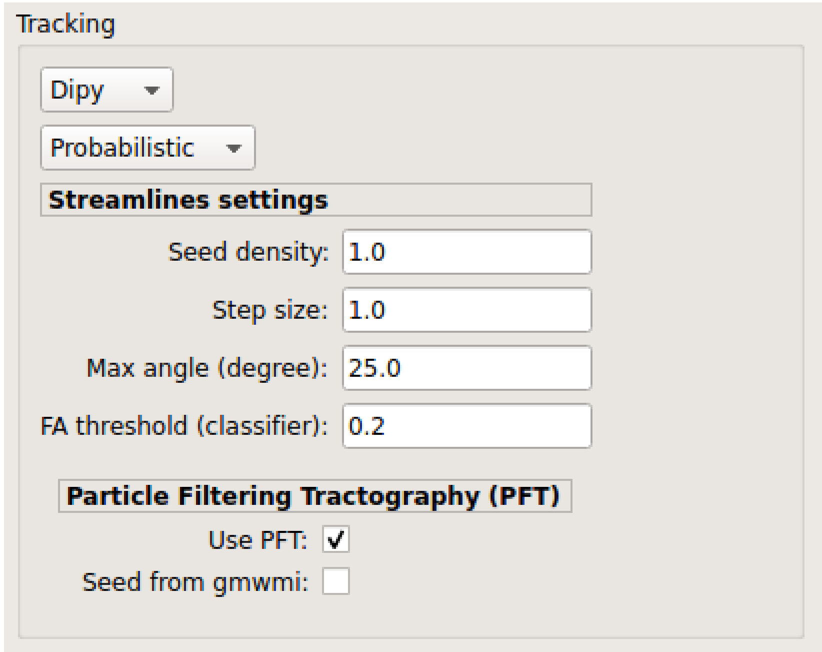
Deterministic tractography (SD_STREAM) performed on single tensor or CSD reconstruction

- Probabilistic tractography:



Probabilistic tractography (iFOD2) performed on SHORE or CSD reconstruction

- Probabilistic particle filtering tractography (PFT):

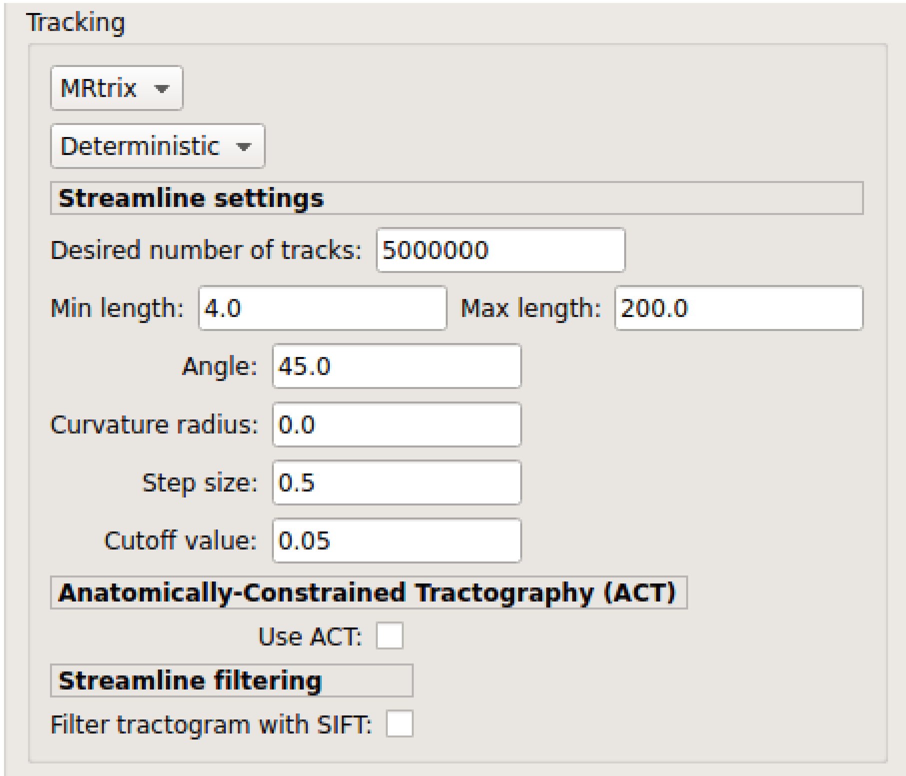


Probabilistic PFT tracking performed on SHORE or CSD reconstruction. Seeding from the gray matter / white matter interface is possible.

Note: We noticed a shift of the center of tractograms obtained by dipy. As a result, tractograms visualized in TrackVis are not commonly centered despite the fact that the tractogram and the ROIs are properly aligned.

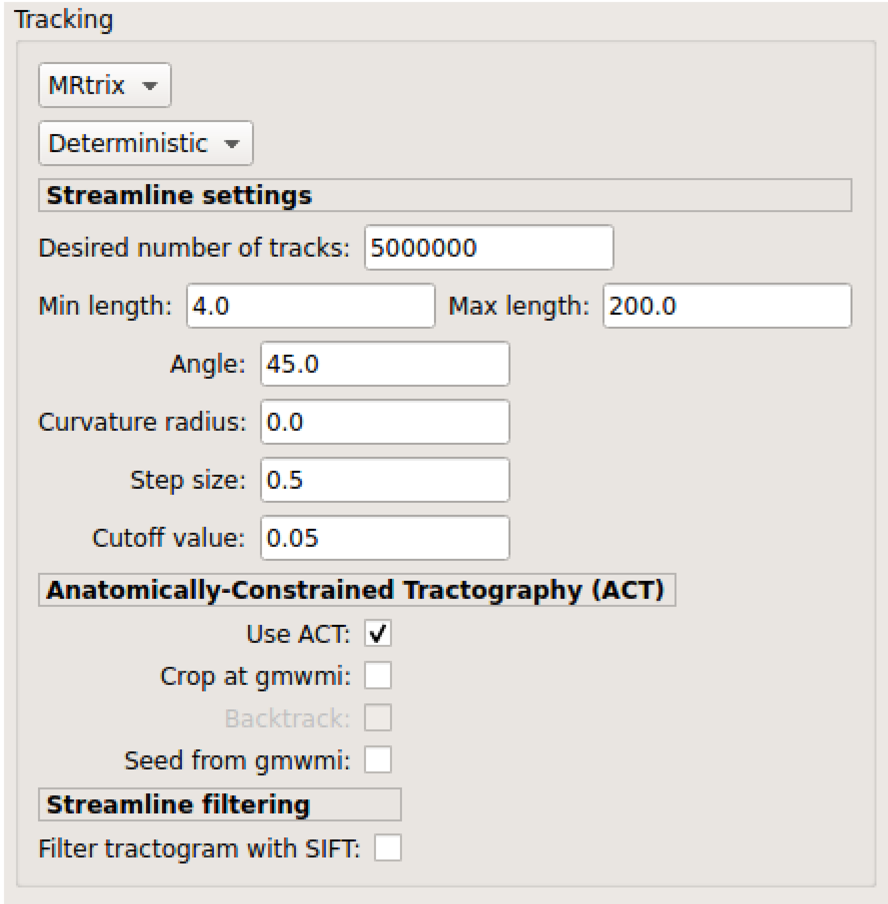
MRtrix: perform deterministic and probabilistic fiber tracking as well as anatomically-constrained tractography. ROI dilation is required to map brain connections when the tracking only operates in the white matter.

- Deterministic tractography:



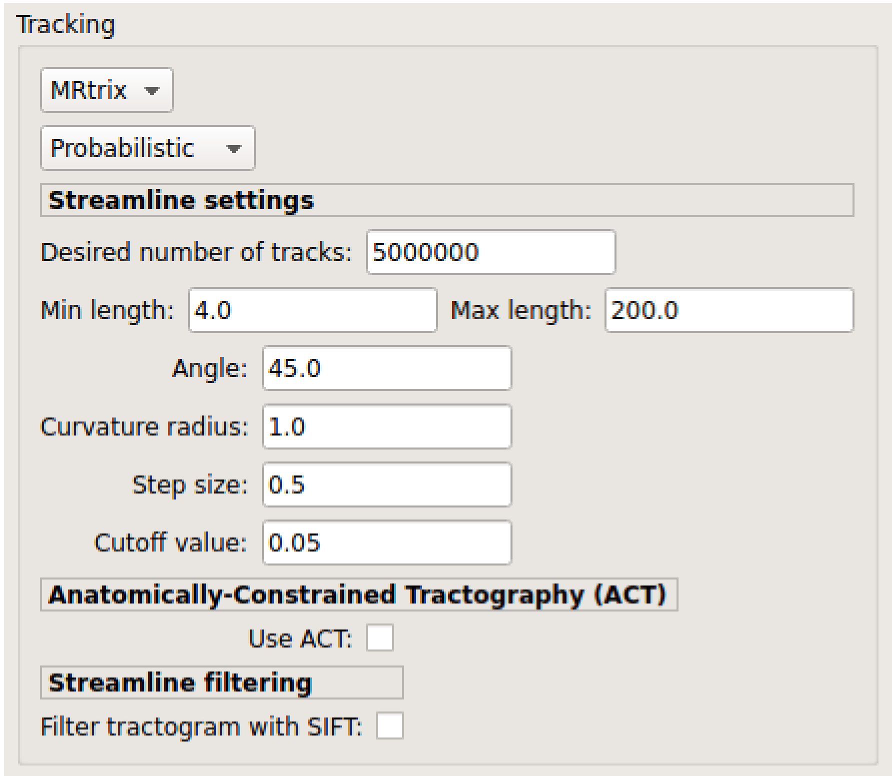
Deterministic tractography (SD_STREAM) performed on single tensor or CSD reconstruction

- Deterministic anatomically-constrained tractography (ACT):



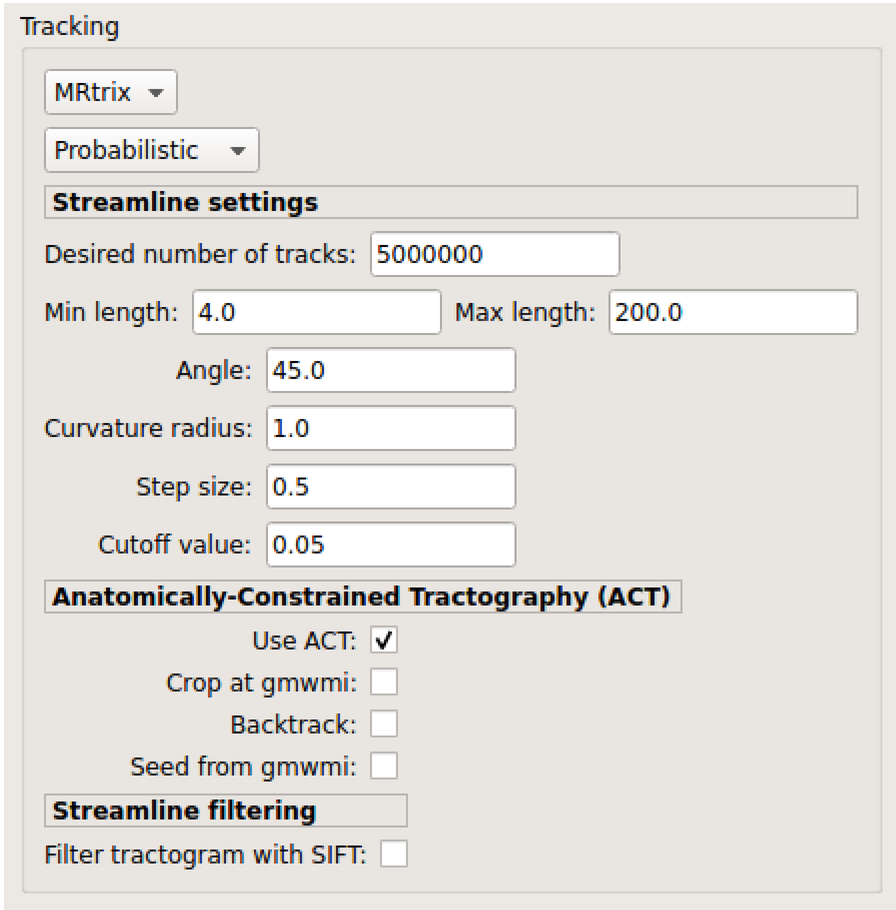
Deterministic ACT tracking performed on single tensor or CSD reconstruction. Seeding from the gray matter / white matter interface is possible. Backtrack option is not available in deterministic tracking.

- Probabilistic tractography:



Probabilistic tractography (iFOD2) performed on SHORE or CSD reconstruction

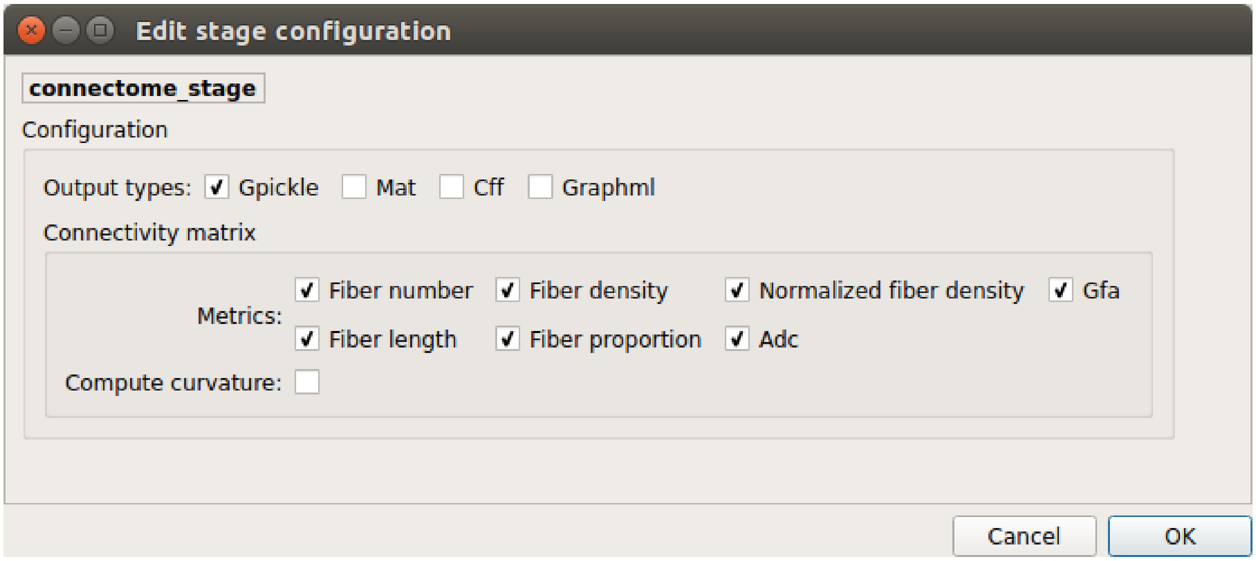
- Probabilistic anatomically-constrained tractography (ACT):



Probabilistic ACT tracking performed on SHORE or CSD reconstruction. Seeding from the gray matter / white matter interface is possible.

Connectome

Compute fiber length connectivity matrices. If DTI data is processed, FA additional map is computed. In case of DSI, additional maps include GFA and RTOP. In case of MAP-MRI, additional maps are RTPP, RTOP, …



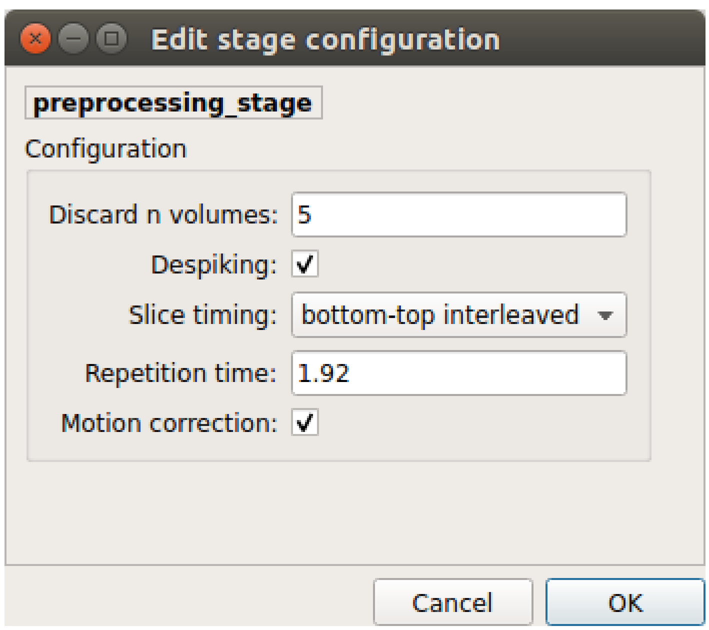
Output types

Select in which formats the connectivity matrices should be saved.

FMRI pipeline stages

Preprocessing

Preprocessing refers to processing steps prior to registration. It includes discarding volumes, despiking, slice timing correction and motion correction for fMRI (BOLD) data.



Discard n volummes

Discard n volumes from further analysis

Despiking

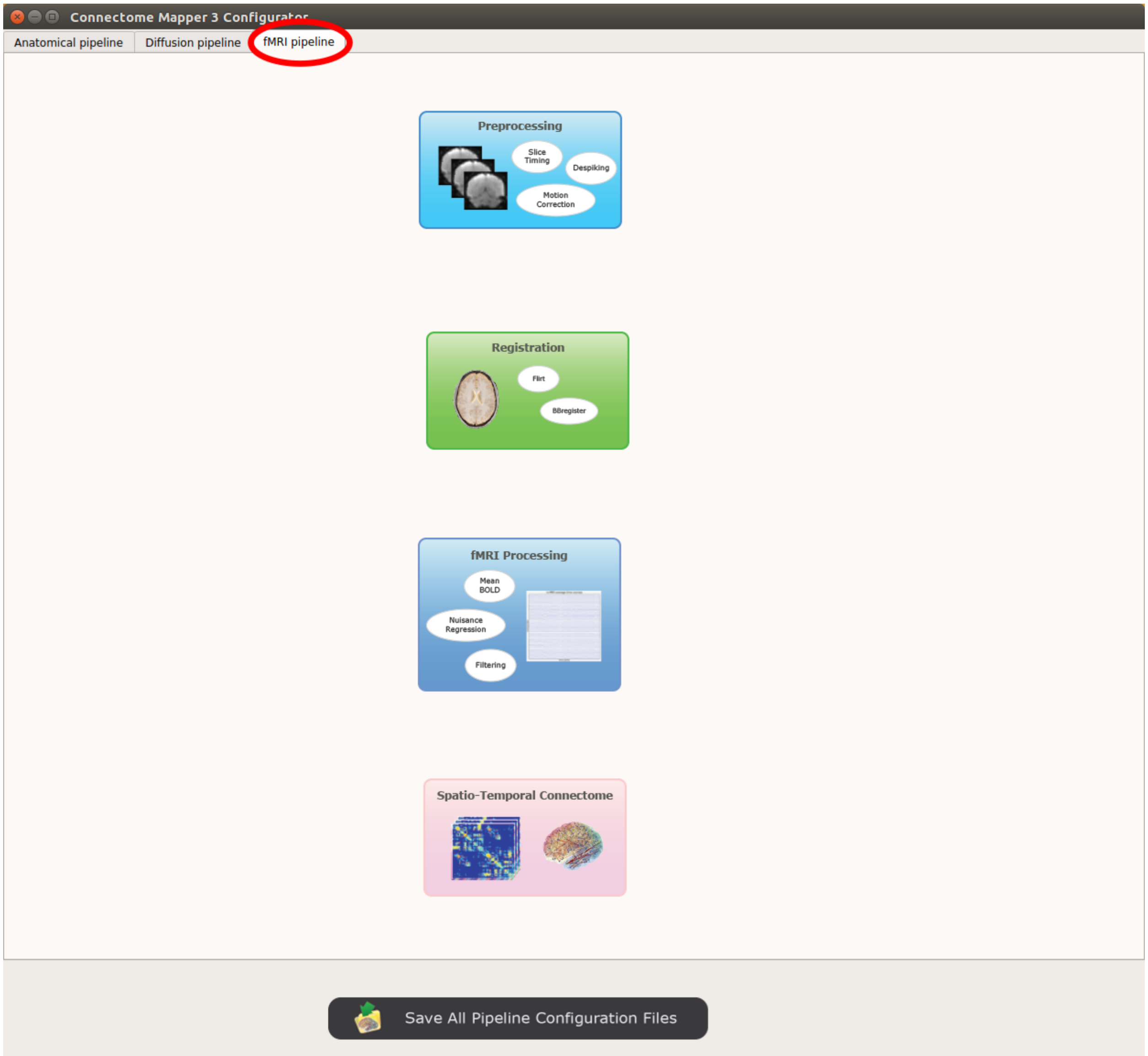


Fig. 6: Panel for configuration of fMRI pipeline stages

Perform despiking of the BOLD signal using AFNI.

Slice timing and Repetition time

Perform slice timing correction using FSL's slicetimer.

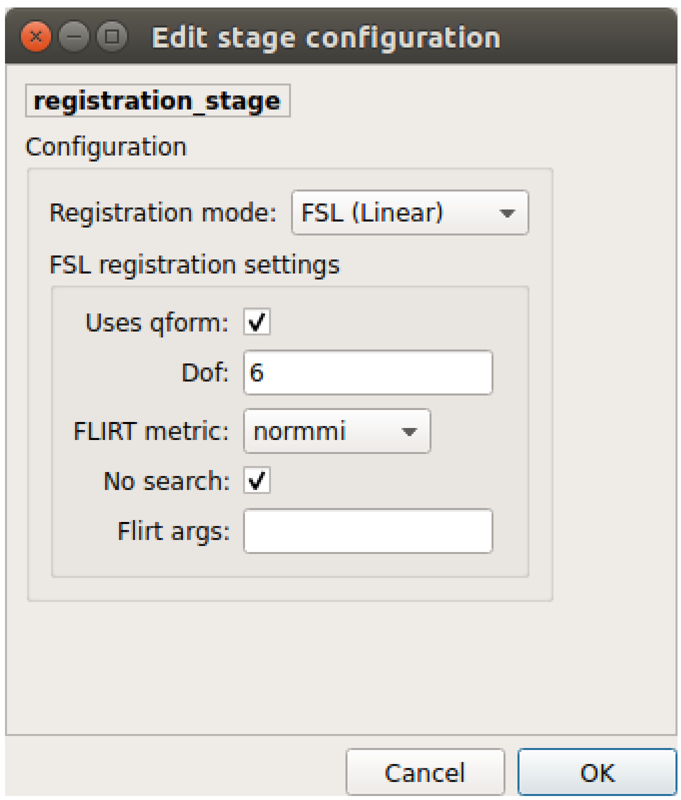
Motion correction

Align BOLD volumes to the mean BOLD volume using FSL's MCFLIRT.

Registration

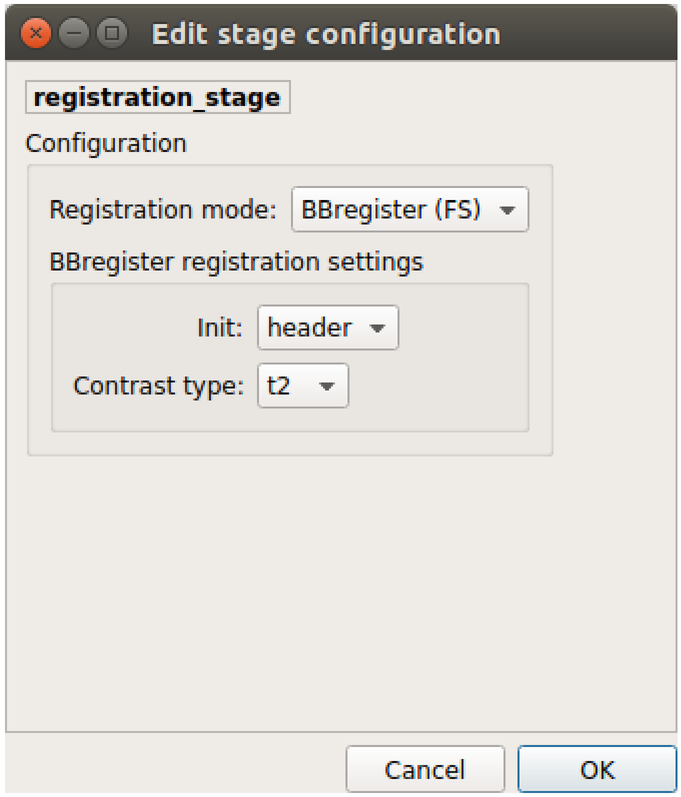
Registration mode

- FSL (Linear):



Perform linear registration from T1 to mean BOLD using FSL's flirt.

- BBregister (FS)



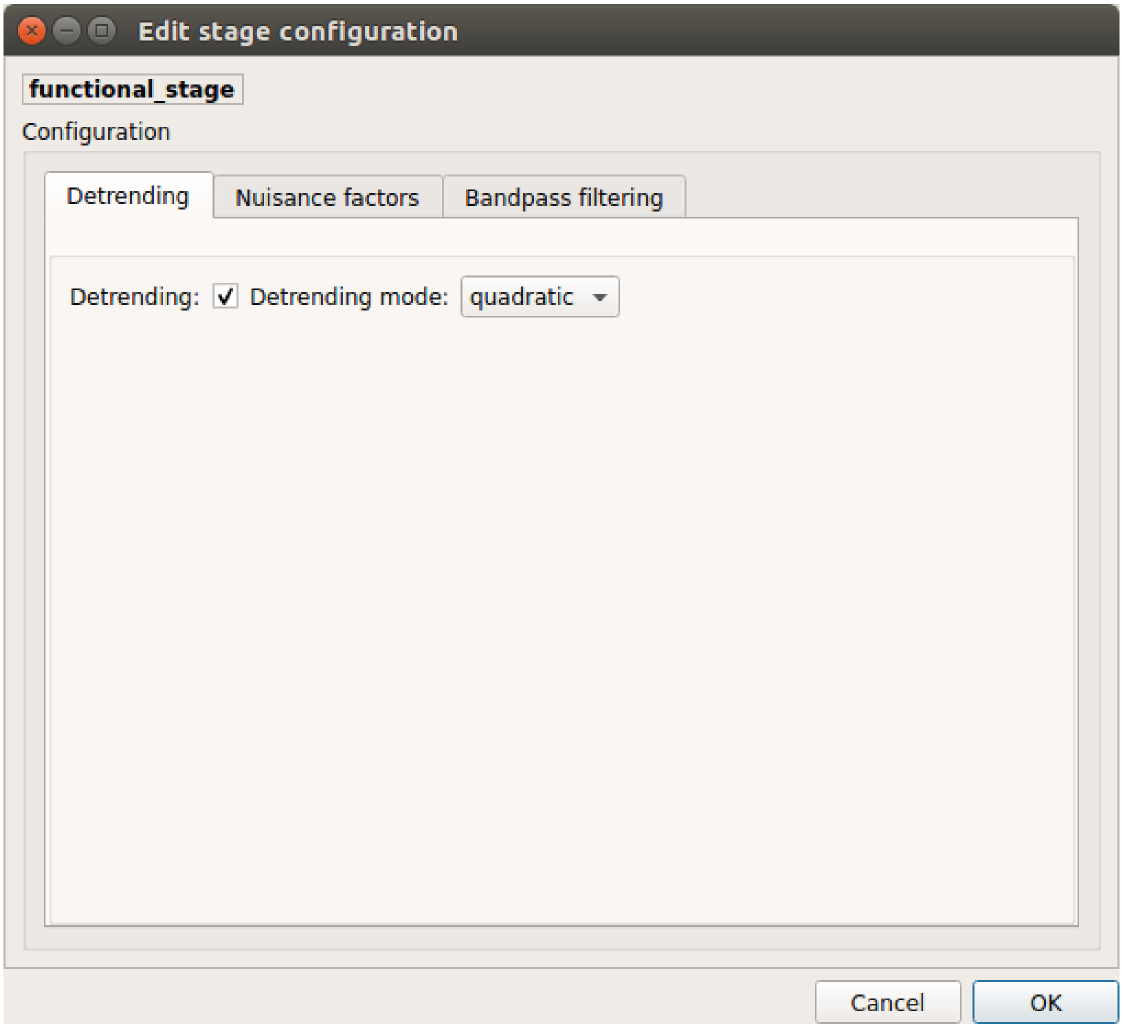
Perform linear registration using Freesurfer BBregister tool from T1 to mean BOLD via T2.

Warning: development in progress

fMRI processing

Performs detrending, nuisance regression, bandpass filteringdiffusion reconstruction and local deterministic or probabilistic tractography based on several tools. ROI dilation is required to map brain connections when the tracking only operates in the white matter.

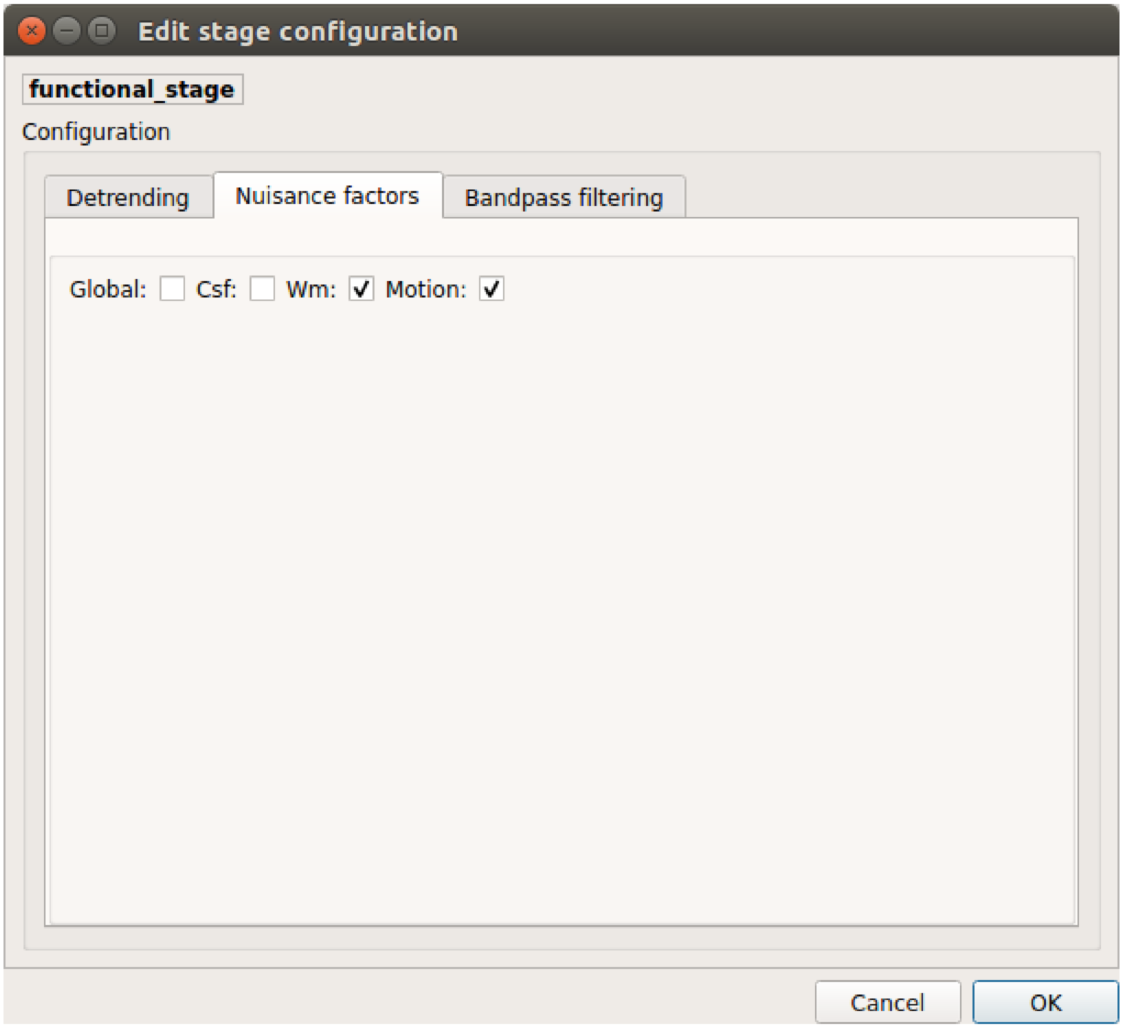
Detrending



Detrending of BOLD signal using:

1. *linear* trend removal algorithm provided by the `scipy` library
2. *quadratic* trend removal algorithm provided by the `obspy` library

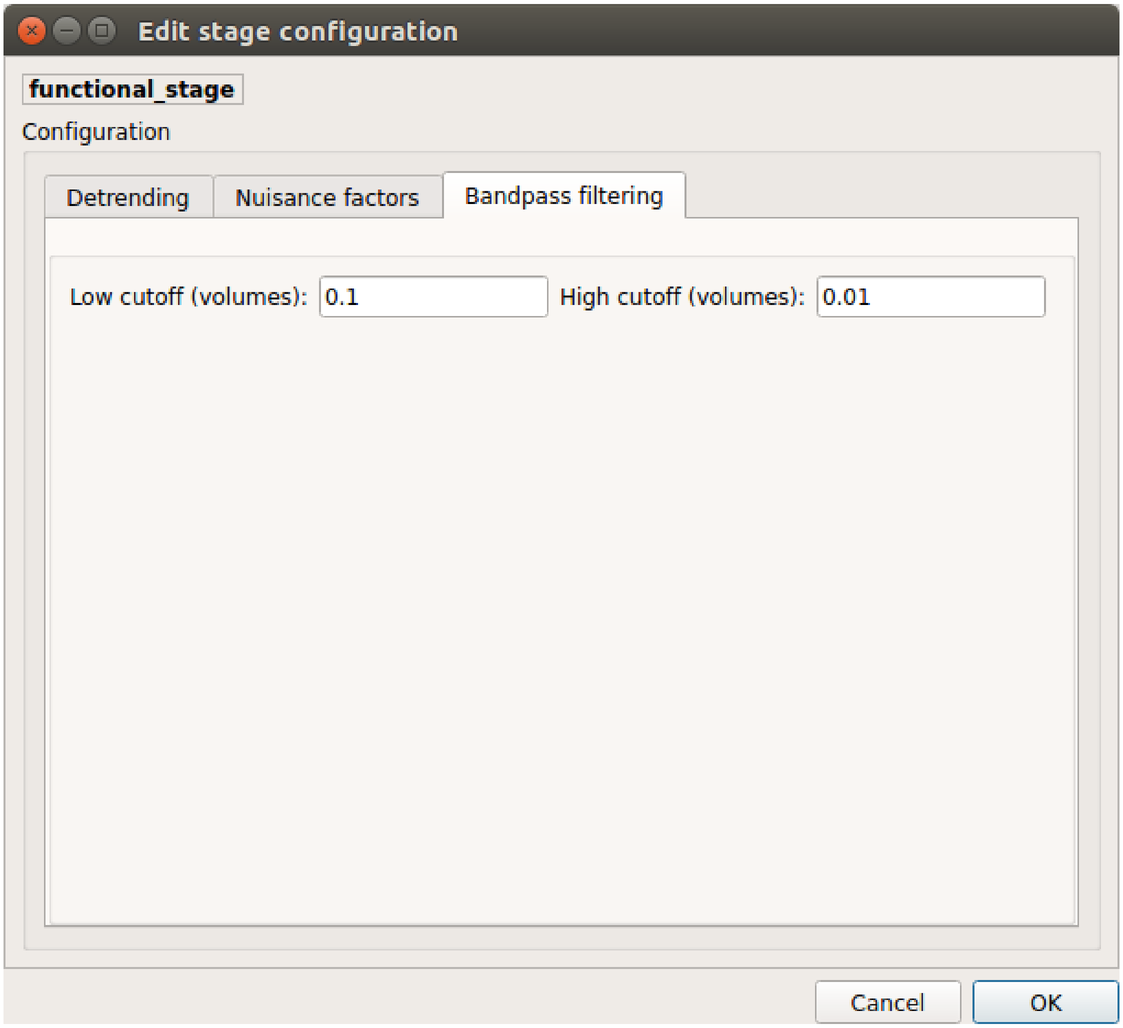
Nuisance regression



A number of options for removing nuisance signals is provided. They consist of:

1. *Global signal* regression
2. *CSF* regression
3. *WM* regression
4. *Motion parameters* regression

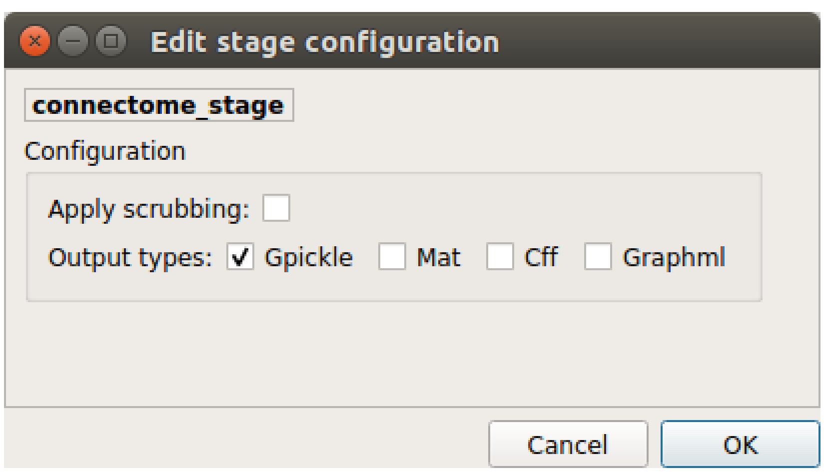
Bandpass filtering



Perform bandpass filtering of the time-series using FSL's slicetimer

Connectome

Computes ROI-averaged time-series and the correlation connectivity matrices.



Output types

Select in which formats the connectivity matrices should be saved.

EEG pipeline stages

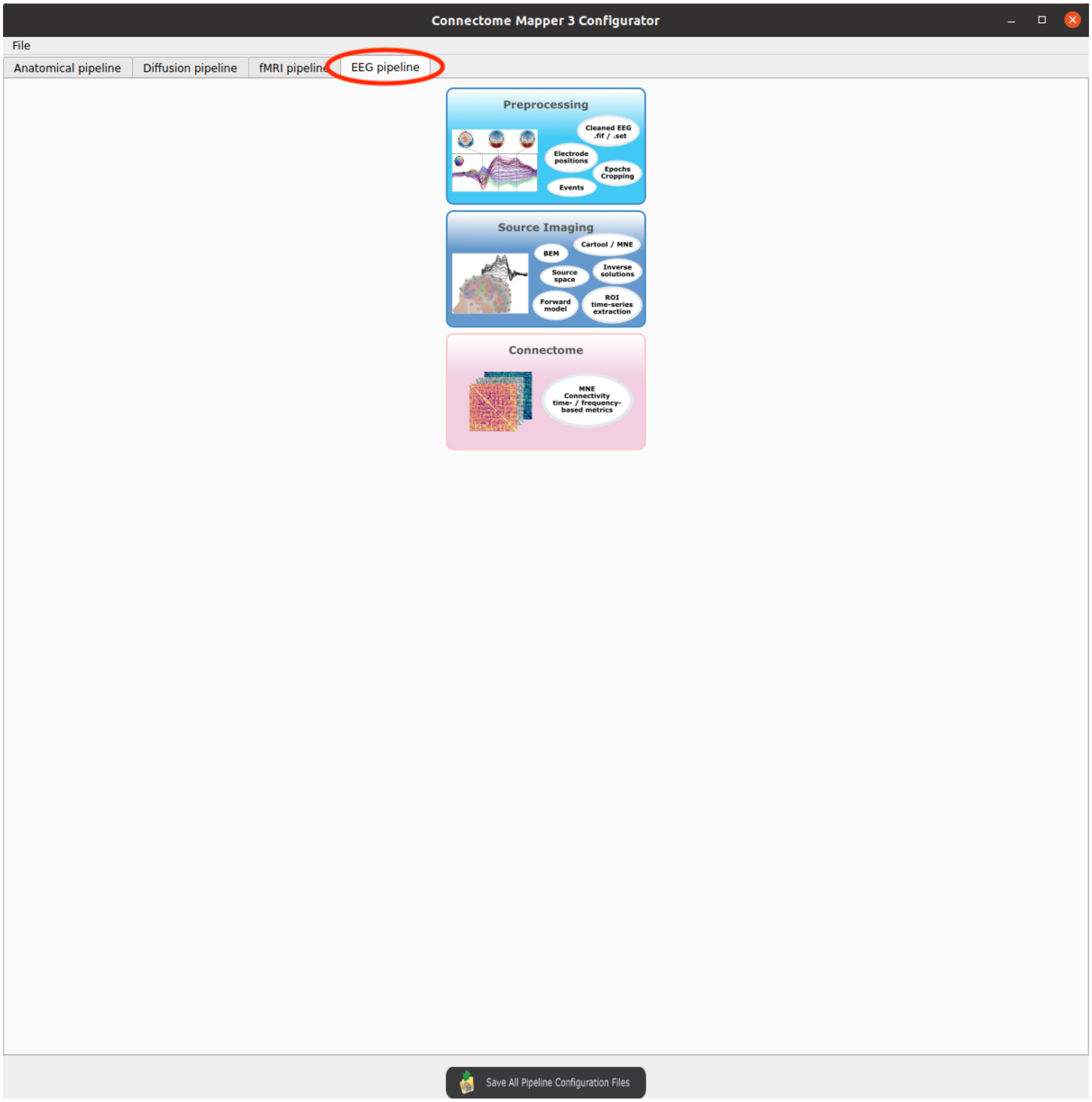


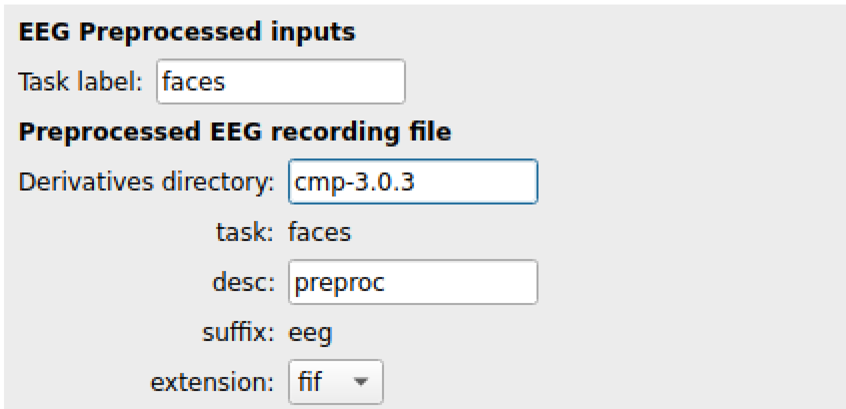
Fig. 7: Panel for configuration of EEG pipeline stages

EEG Preprocessing

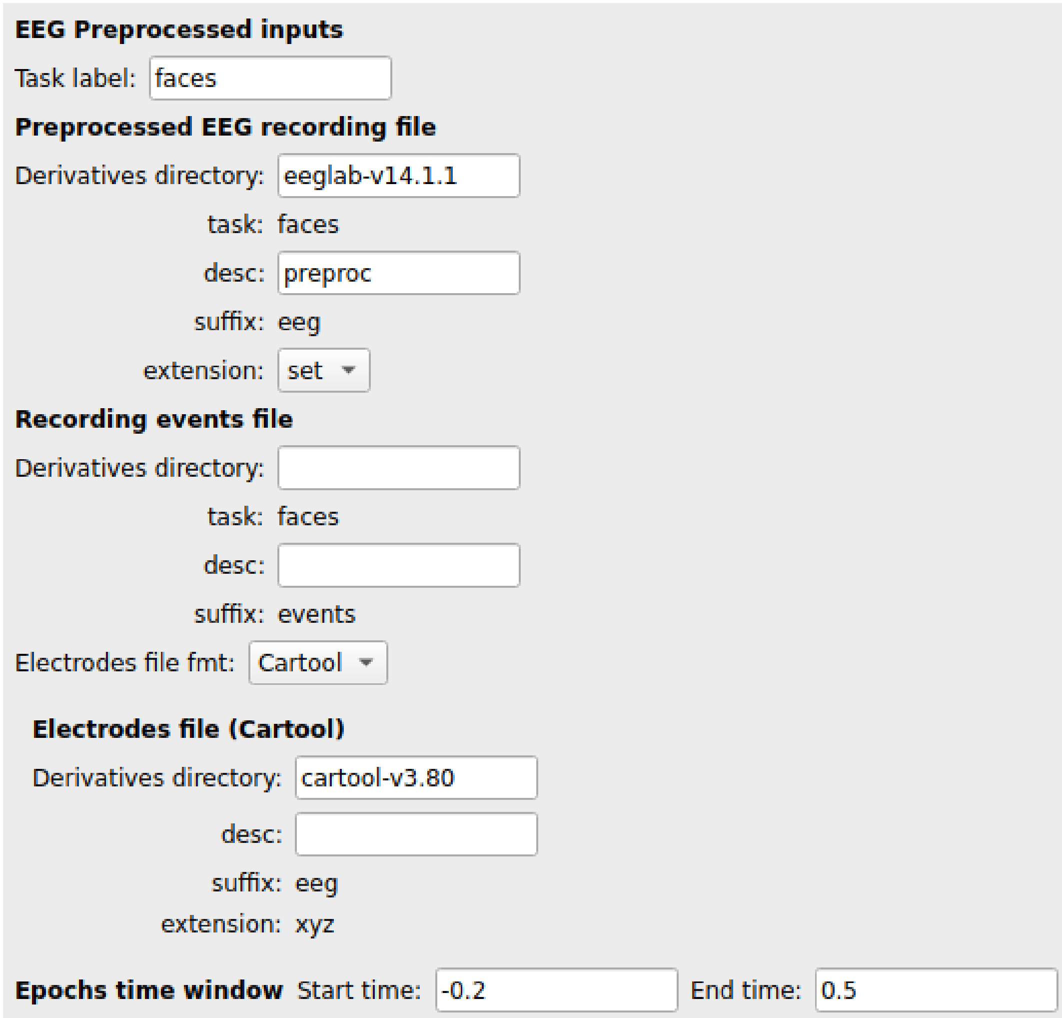
EEG Preprocessing refers to steps that loads, crop, and save preprocessed EEG epochs data of a given task in `fif` format, the harmonized format used further in the pipeline.

EEG data can be provided as:

1. A `mne.Epochs` object already saved in `fif` format:



2. A set of the following files and parameters:

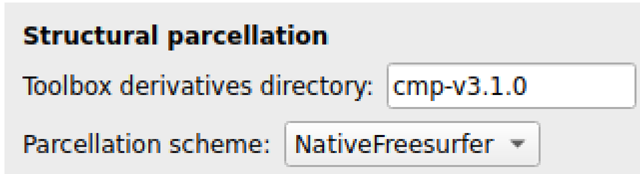


- *Preprocessed EEG recording*: store the Epochs * Electrodes dipole time-series in eeglab `set` format
- *Recording events file* in BIDS *_events.tsv format: describe timing and other properties of events recorded during the task
- *Electrodes file* file in **`BIDS *_electrodes.tsv format`_** or in Cartool *.xyz format: store the electrode coordinates
- *Epochs time window*: relative start and end time to crop the epochs

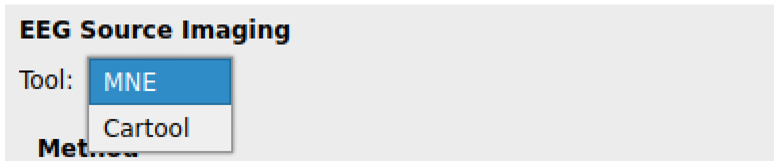
EEG Source Imaging

EEG Source Imaging refers to the all the steps necessary to obtain the inverse solutions and extract ROI time-series for a given parcellation scheme.

- *Structural parcellation*: specify the cmp derivatives directory, the parcellation scheme, and the scale (for Lausanne 2018) to retrieve the parcellation files

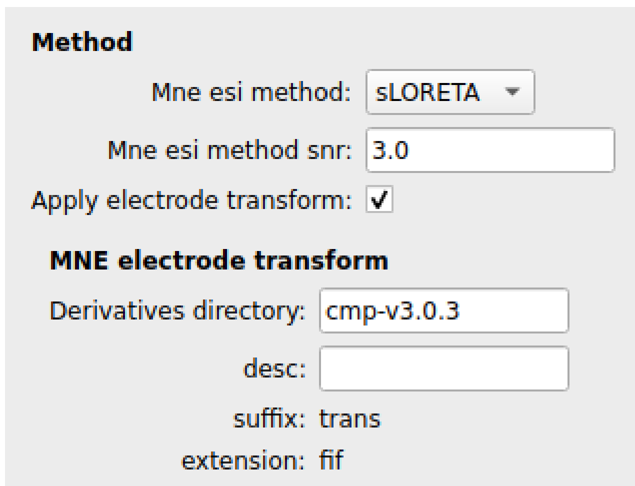


- *Tool*: CMP3 can either leverage MNE to compute the inverse solutions or take inverse solutions already pre-computed with Cartool as input.



  – *MNE*

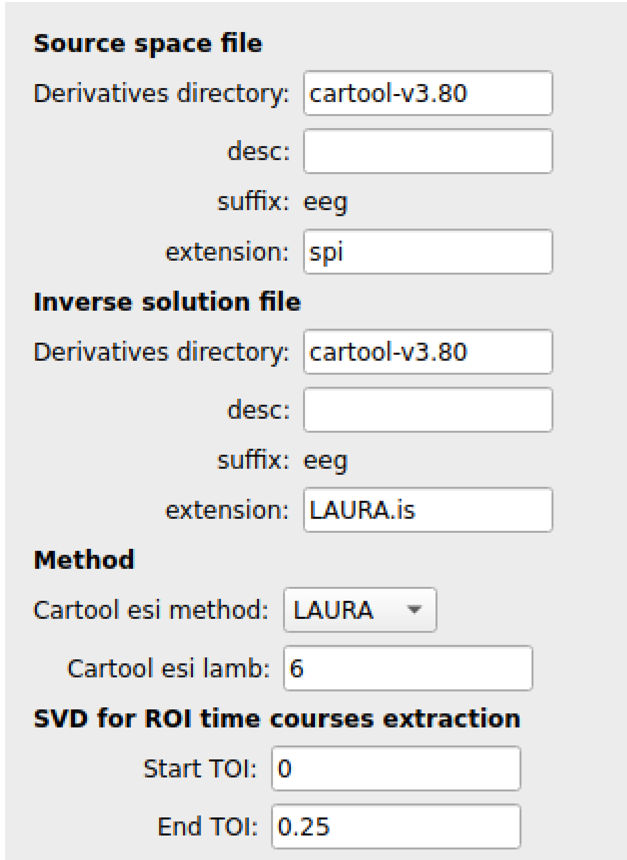
    If `MNE` is selected, all steps necessary to reconstruct the inverse solutions are performed by leveraging MNE. In this case, the following files and parameters need to be provided:



    * *MNE ESI method*: Method to compute the inverse solutions
    * *MNE ESI method SNR*: SNR level used to regularize the inverse solutions
    * *MNE electrode transform*: Additional transform in MNE `trans.fif` format to be applied to electrode coordinates when *Apply electrode transform* is enabled

  – *Cartool*

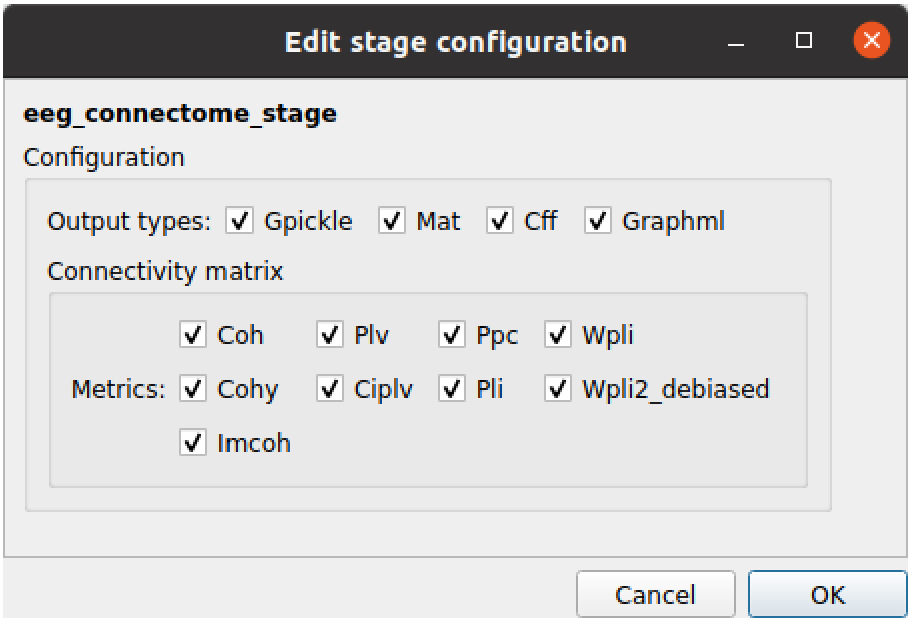
    If `Cartool` is selected, the following files (generated by this tool) and parameters need to be provided:



- *Source space file*: `*.spi` text file with 3D-coordinates (x, y and z-dimension) with possible solution points necessary to obtain the sources or generators of ERPs
- *Inverse solution file*: `*.is` binary file that includes number of electrodes and solution points
- *Cartool esi method*: Method used to compute the inverse solutions (*Cartool esi method*)
- *Cartool esi lamb*: Regularization level of inverse solutions
- *SVD for ROI time-courses extraction*: Start and end TOI parameters for the SVD algorithm that extract single ROI time-series from dipoles.

EEG Connectome

Computes frequency- and time-frequency-domain connectivity matrices with MNE Spectral Connectivity.



Output types

Select in which formats the connectivity matrices should be saved.

Save the configuration files

You can save the pipeline stage configuration files in two different way:

1. You can save all configuration files at once by clicking on the Save All Pipeline Configuration Files. This will save automatically the configuration file of the anatomical / diffusion / fMRI pipeline to <bids_dataset>/code/ref_anatomical_config.ini / <bids_dataset>/code/ref_diffusion_config.ini / <bids_dataset>/code/ref_fMRI_config.ini, <bids_dataset>/code/ref_EEG_config.ini respectively.
2. You can save individually each of the pipeline configuration files and edit its filename in the File menu (File -> Save anatomical/diffusion/fMRI/EEG configuration file as…)

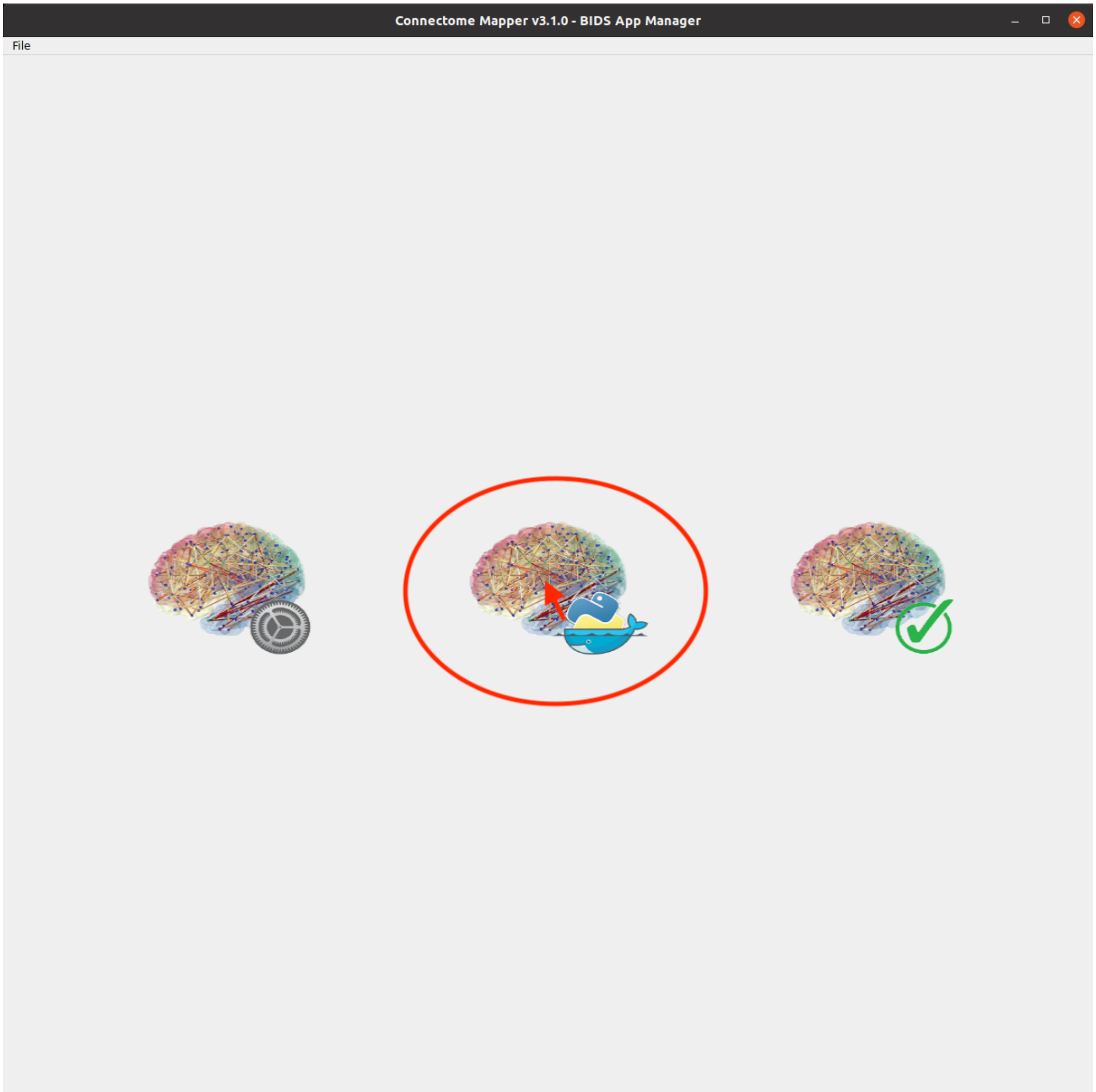
Nipype

Connectome Mapper relies on Nipype. All intermediate steps for the processing are saved in the corresponding <bids_dataset/derivatives>/nipype/sub-<participant_label>/<pipeline_name>/<stage_name> stage folder (See *Nipype workflow outputs* for more details).

5.4.5 Run the BIDS App

Start the Connectome Mapper BIDS App GUI

- From the main window, click on the middle button to start the Connectome Mapper BIDS App GUI.



- The window of the Connectome Mapper BIDS App GUI will appear, which will help you in setting up and launching the BIDS App run.

Run configuration

- Select the output directory for data derivatives

- Select the subject labels to be processed

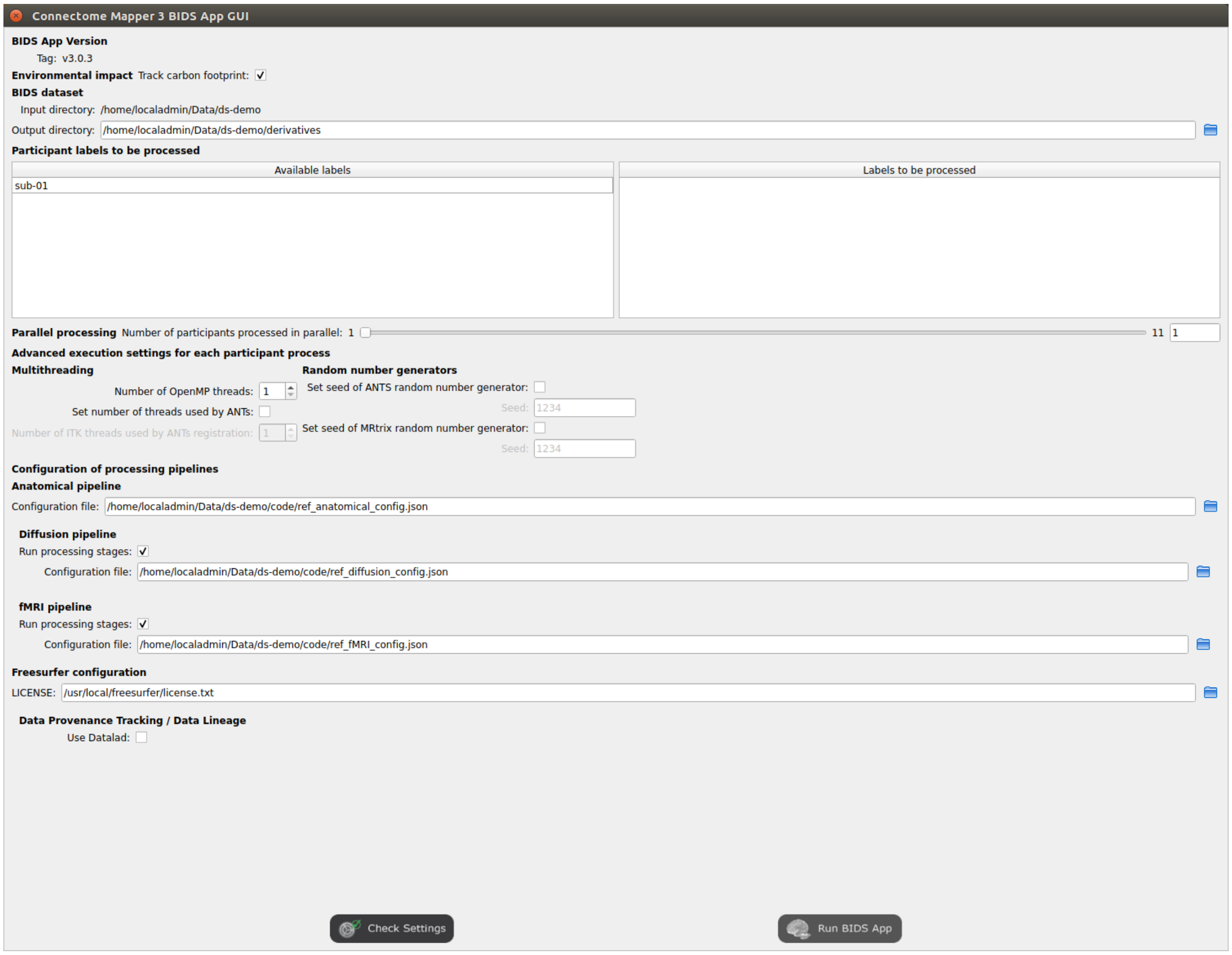
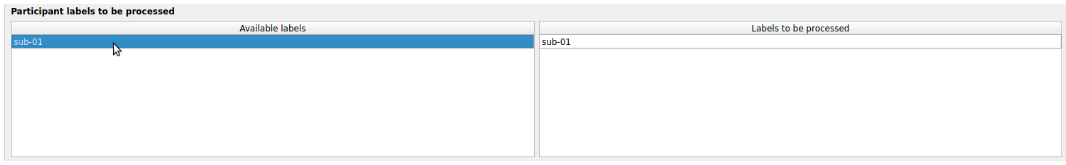
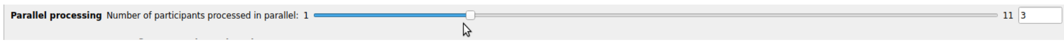


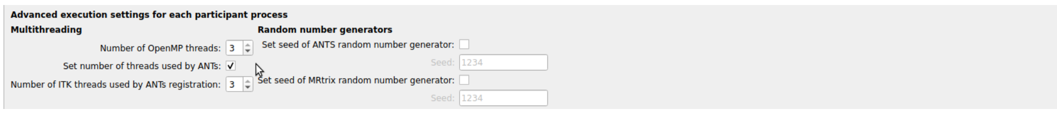
Fig. 8: Window of the Connectome Mapper BIDS App GUI



- Tune the number of subjects to be processed in parallel



- Tune the advanced execution settings for each subject process. This include finer control on the number of threads used by each process as well as on the seed value of ANTs and MRtrix random number generators.



Important: Make sure the number of threads multiplied by the number of subjects being processed in parallel do not exceed the number of CPUs available on your system.

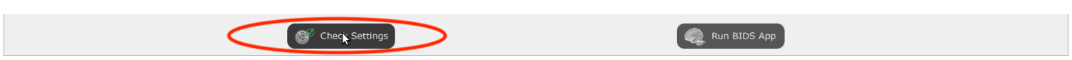
- Check/Uncheck the pipelines to be performed

Note: The list of pipelines might vary as it is automatically updated based on the availability of raw diffusion MRI, resting-state fMRI, and preprocessed EEG data.

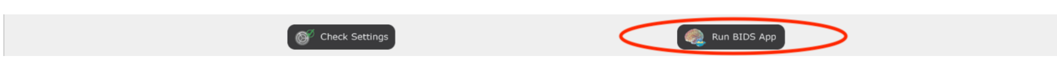
- Specify your Freesurfer license

Note: Your Freesurfer license will be copied to your dataset directory as `<bids_dataset>/code/license.txt` which will be mounted inside the BIDS App container image.

- When the run is set up, you can click on the `Check settings` button.



- If the setup is complete and valid, this will enable the `Run BIDS App` button.



You are ready to launch the BIDS App run!

Execute the BIDS App

- Click on the `Run BIDS App` button to execute the BIDS App

- You can see the complete `docker run` command generated by the Connectome Mapper BIDS App GUI from the terminal output such as in this example

```
Start BIDS App
> FreeSurfer license copy skipped as it already exists (BIDS App Manager)
> Datalad available: True
... BIDS App execution command: ['docker', 'run', '-it', '--rm', '-v', '/
↪home/localadmin/Data/ds-demo:/bids_dir', '-v', '/home/localadmin/Data/ds-
↪demo/derivatives:/output_dir', '-v', '/usr/local/freesurfer/license.txt:/
↪bids_dir/code/license.txt', '-v', '/home/localadmin/Data/ds-demo/code/ref_
↪anatomical_config.ini:/code/ref_anatomical_config.ini', '-v', '/home/
↪localadmin/Data/ds-demo/code/ref_diffusion_config.ini:/code/ref_diffusion_
↪config.ini', '-v', '/home/localadmin/Data/ds-demo/code/ref_fMRI_config.
↪ini:/code/ref_fMRI_config.ini', '-u', '1000:1000', 'sebastientourbier/
↪connectomemapper-bidsapp:v3.0.3', '/bids_dir', '/output_dir', 'participant
↪', '--participant_label', '01', '--anat_pipeline_config', '/code/ref_
↪anatomical_config.ini', '--dwi_pipeline_config', '/code/ref_diffusion_
↪config.ini', '--func_pipeline_config', '/code/ref_fMRI_config.ini', '--fs_
↪license', '/bids_dir/code/license.txt', '--number_of_participants_
↪processed_in_parallel', '1', '--number_of_threads', '3', '--ants_number_
↪of_threads', '3']
> BIDS dataset: /bids_dir
> Subjects to analyze : ['01']
> Set $FS_LICENSE which points to FreeSurfer license location (BIDS App)
  ... $FS_LICENSE : /bids_dir/code/license.txt
  * Number of subjects to be processed in parallel set to 1 (Total of cores␣
↪available: 11)
  * Number of parallel threads set to 10 (total of cores: 11)
  * OMP_NUM_THREADS set to 3 (total of cores: 11)
  * ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS set to 3
Report execution to Google Analytics.
Thanks to support us in the task of finding new funds for CMP3 development!
> Sessions to analyze : ['ses-01']
> Process subject sub-103818 session ses-01
WARNING: rewriting config file /output_dir/cmp-v3.0.3/sub-01/ses-01/sub-01_
↪ses-01_anatomical_config.ini
... Anatomical config created : /output_dir/cmp-v3.0.3/sub-01/ses-01/sub-01_
↪ses-01_anatomical_config.ini
WARNING: rewriting config file /output_dir/cmp-v3.0.3/sub-01/ses-01/sub-01_
↪ses-01_diffusion_config.ini
... Diffusion config created : /output_dir/cmp-v3.0.3/sub-01/ses-01/sub-01_
↪ses-01_diffusion_config.ini
WARNING: rewriting config file /output_dir/cmp-v3.0.3/sub-01/ses-01/sub-01_
↪ses-01_fMRI_config.ini
... Running pipelines :
        - Anatomical MRI (segmentation and parcellation)
        - Diffusion MRI (structural connectivity matrices)
... cmd : connectomemapper3 --bids_dir /bids_dir --output_dir /output_dir --
↪participant_label sub-01 --session_label ses-01 --anat_pipeline_config /
↪output_dir/cmp-v3.0.3/sub-01/ses-01/sub-01_ses-01_anatomical_config.ini --
↪dwi_pipeline_config /output_dir/cmp-v3.0.3/sub-01/ses-01/sub-01_ses-01_
↪diffusion_config.ini --number_of_threads 3
```

(continues on next page)

(continued from previous page)

```
```

Note: Also, this can be helpful in you wish to design your own batch scripts to call the BIDS App with the correct syntax.

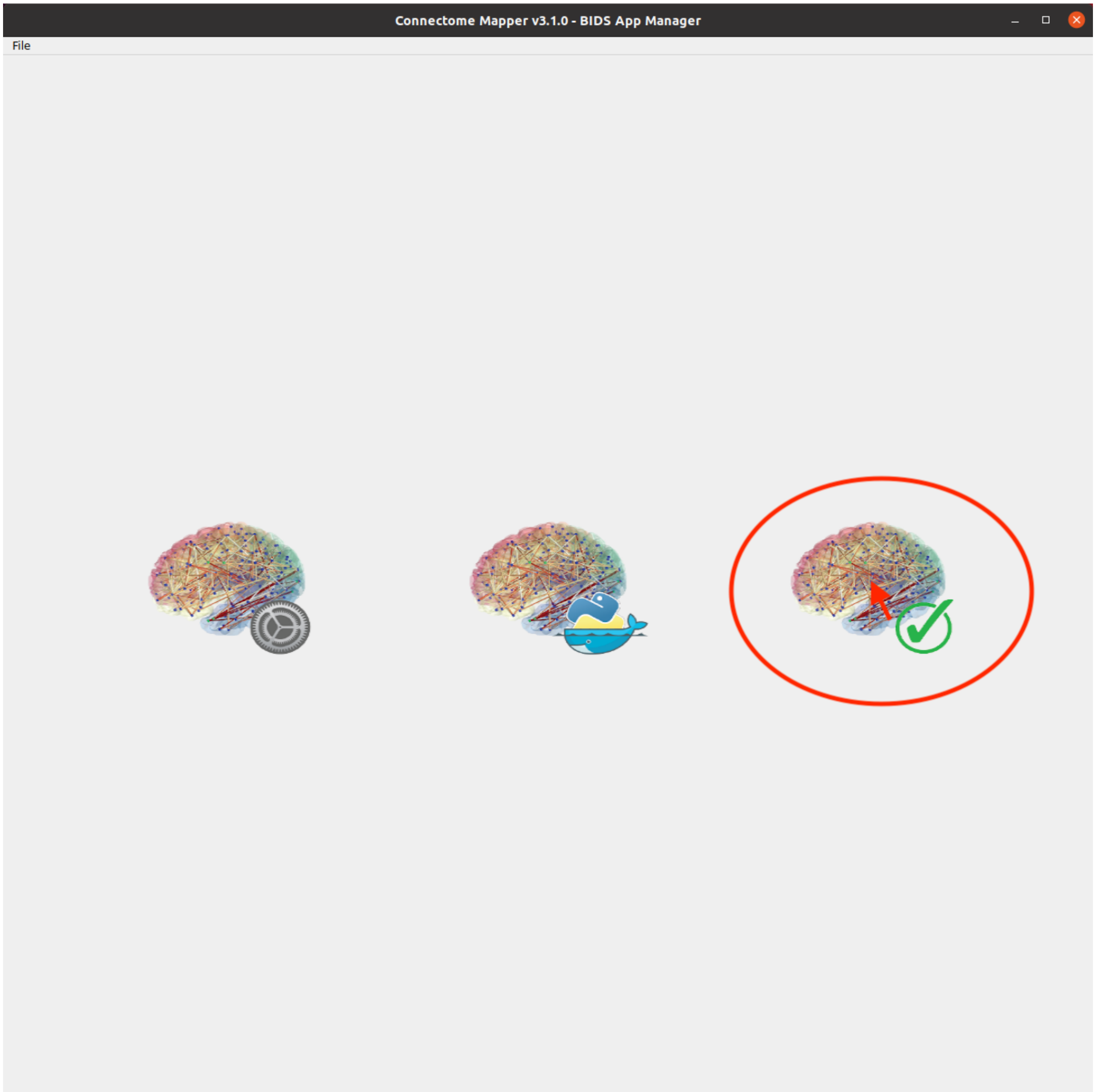
Check progress

For each subject, the execution output of the pipelines are redirected to a log file, written as `<bids_dataset/derivatives>/cmp-v3.X.Y/sub-<subject_label>_log.txt`. Execution progress can be checked by the means of these log files.

5.4.6 Check stages outputs

Start the Inspector Window

- From the main window, click on the right button to start the Inspector Window.



- The Connectome Mapper 3 Inspector Window will appear, which will assist you in inspecting outputs of the different pipeline stages (each pipeline has a tab panel).

Anatomical pipeline stages

- Click on the stage you wish to check the output(s):

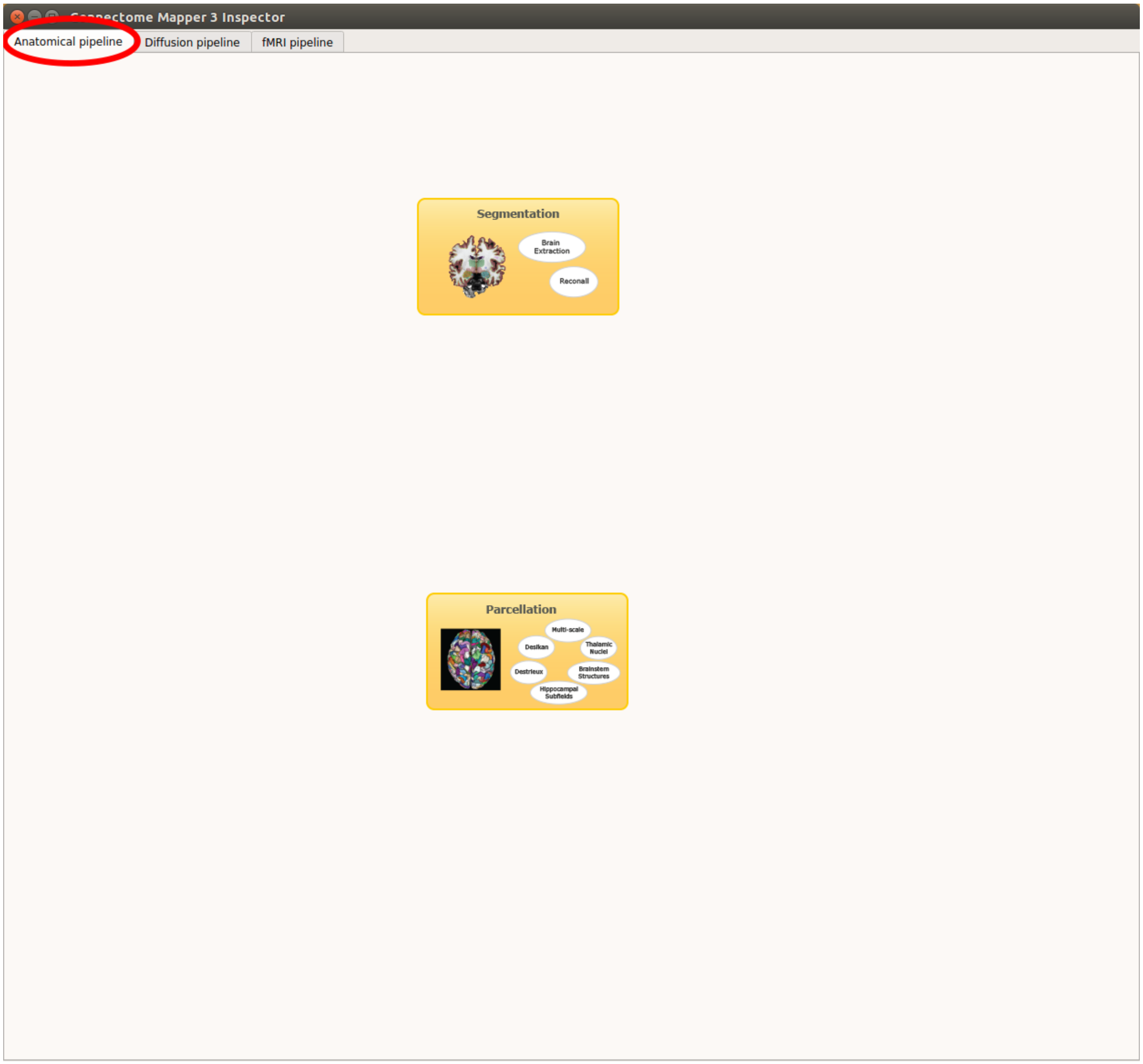
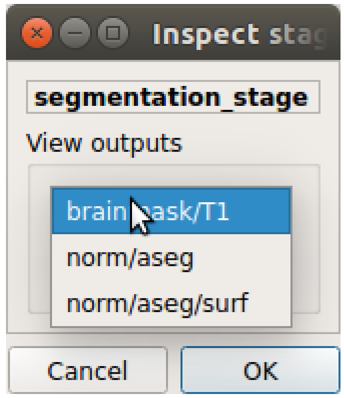


Fig. 9: Panel for output inspection of anatomical pipeline stages

Segmentation

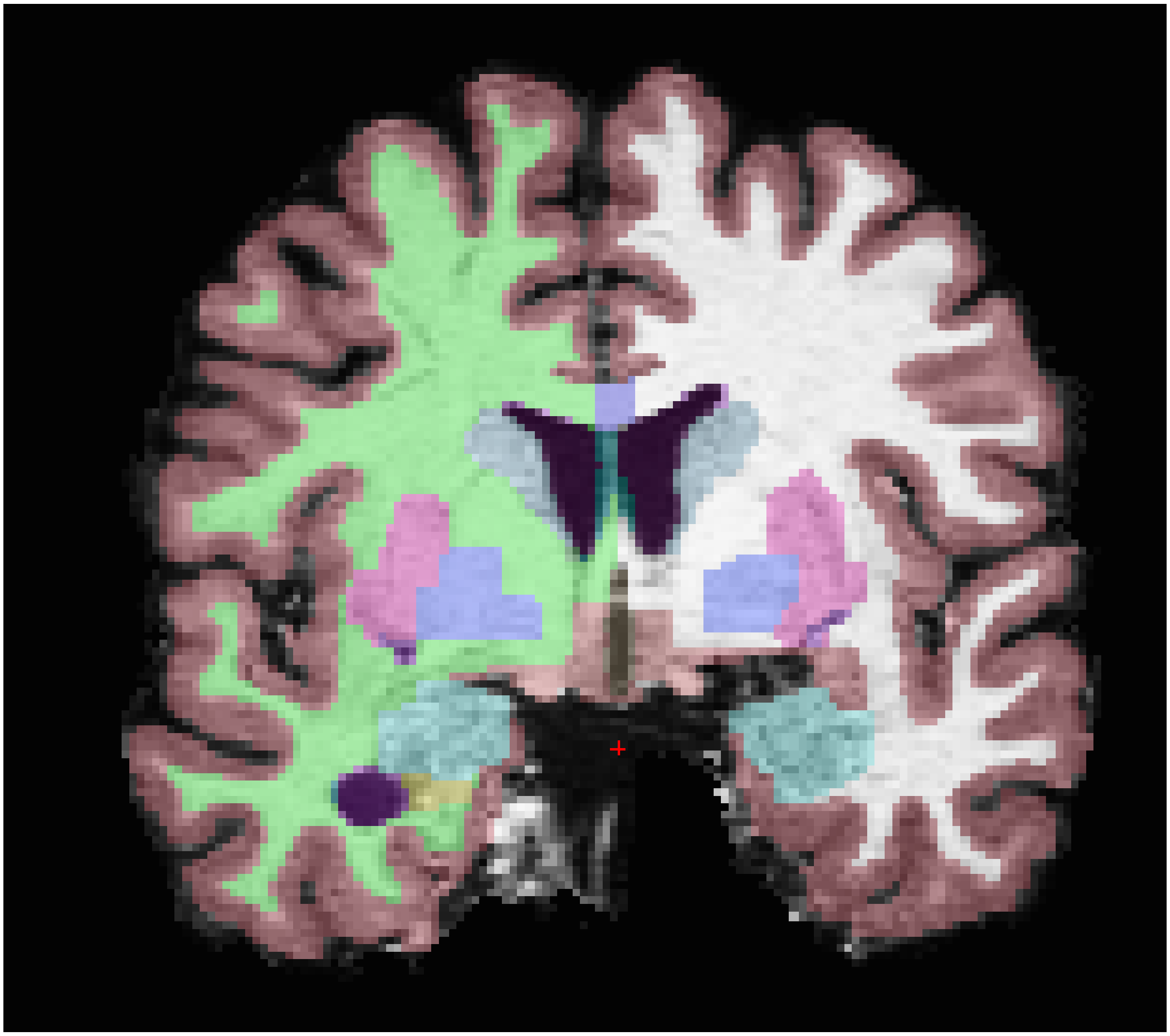
- Select the desired output from the list and click on `view`:



Segmentation results

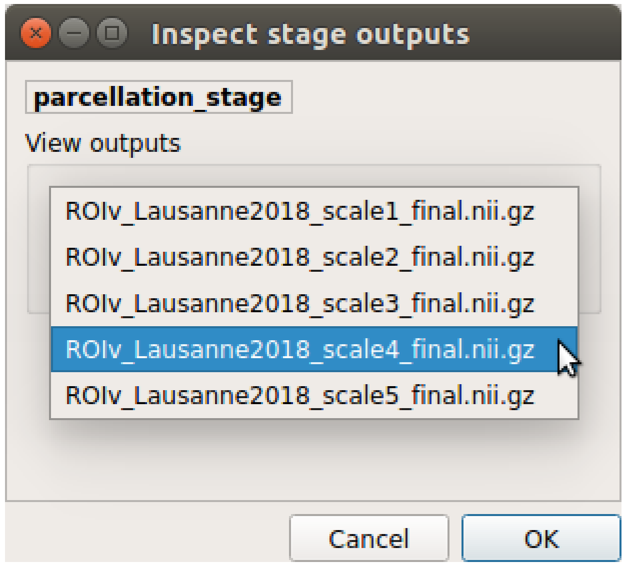
Surfaces extracted using Freesurfer.

T1 segmented using Freesurfer.



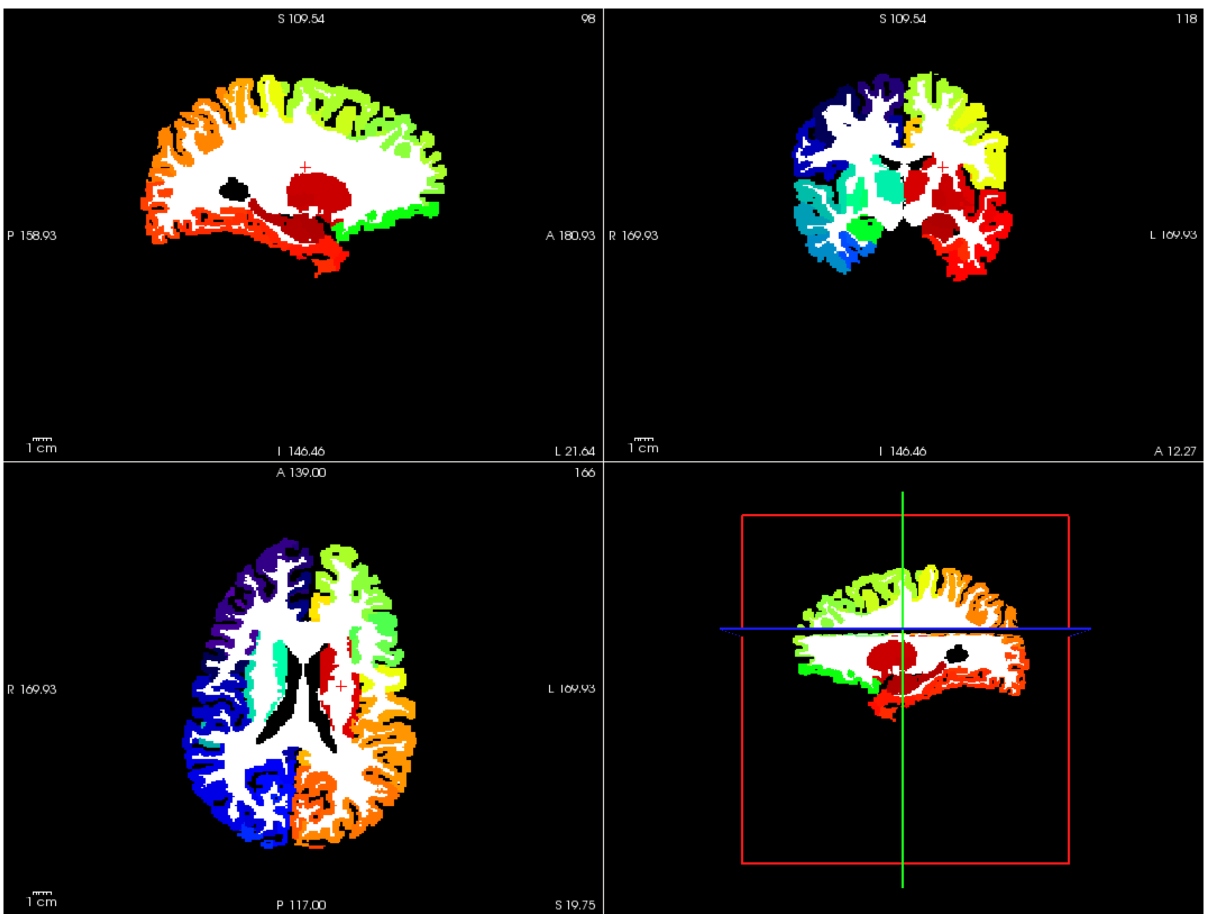
Parcellation

- Select the desired output from the list and click on `view`:



Parcellation results

Cortical and subcortical parcellation are shown with Freeview.



Diffusion pipeline stages

- Click on the stage you wish to check the output(s):

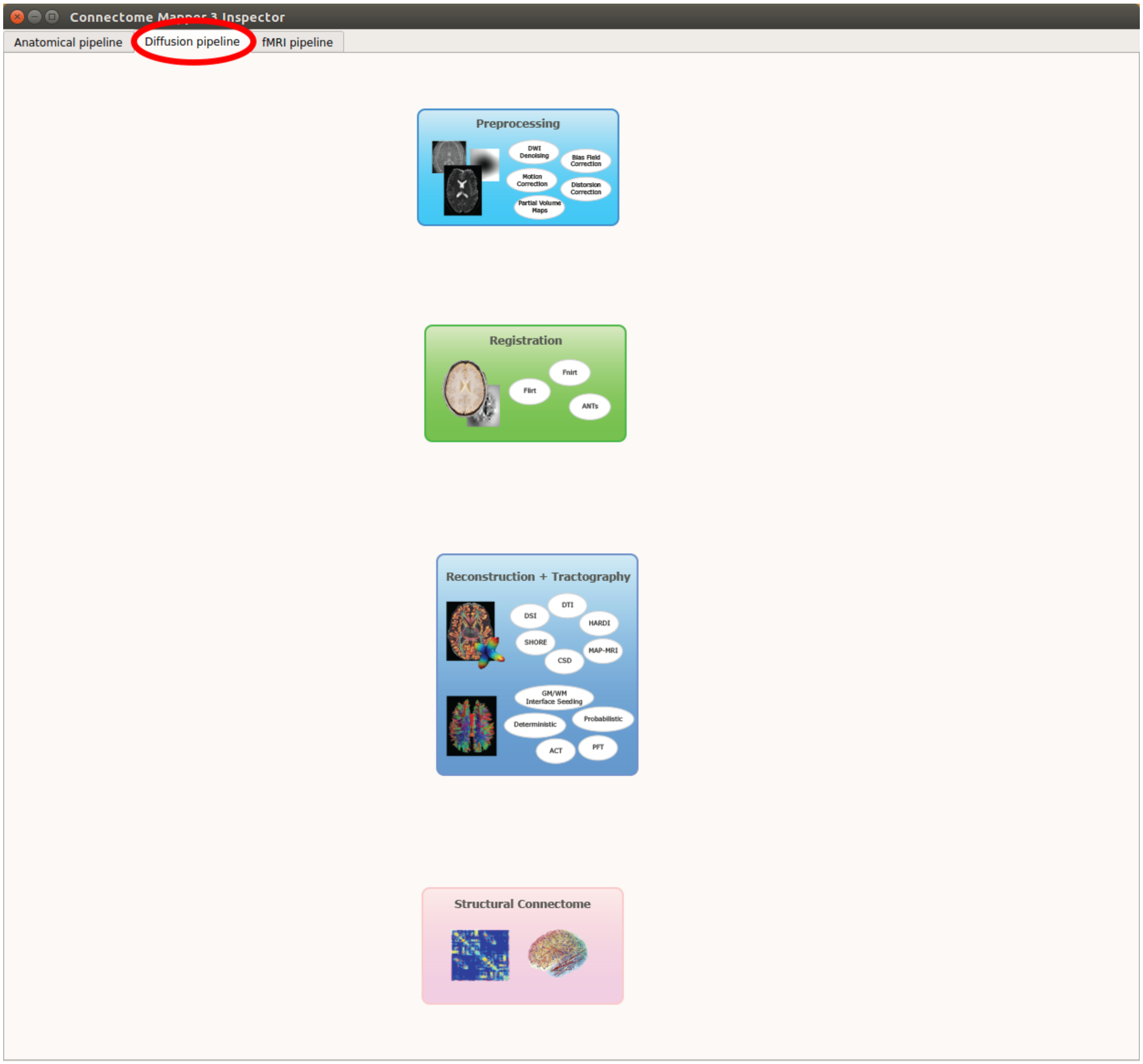
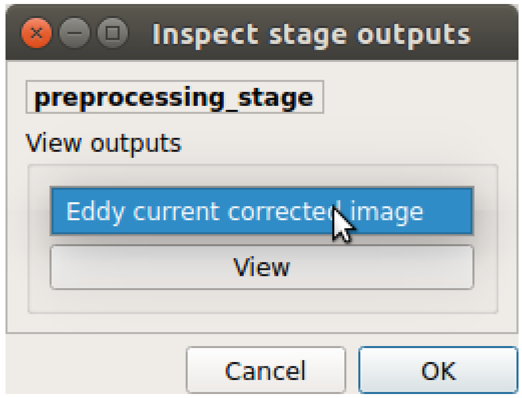


Fig. 10: Panel for output inspection of diffusion pipeline stages

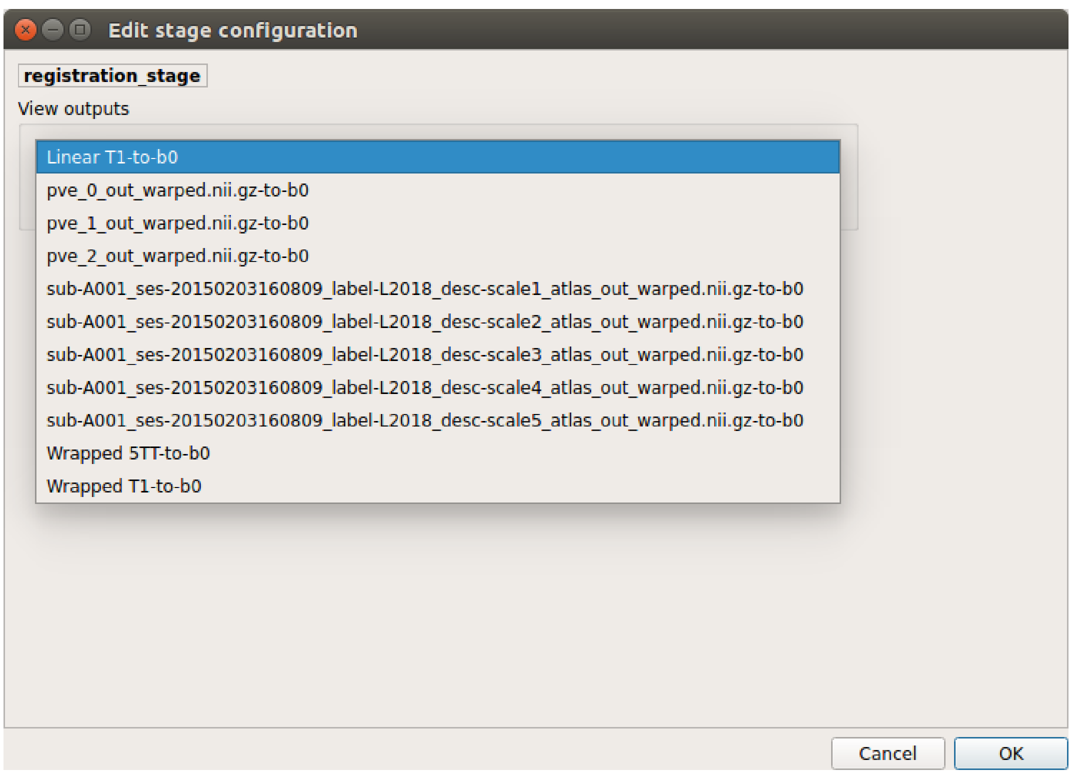
Preprocessing

- Select the desired output from the list and click on `view`:



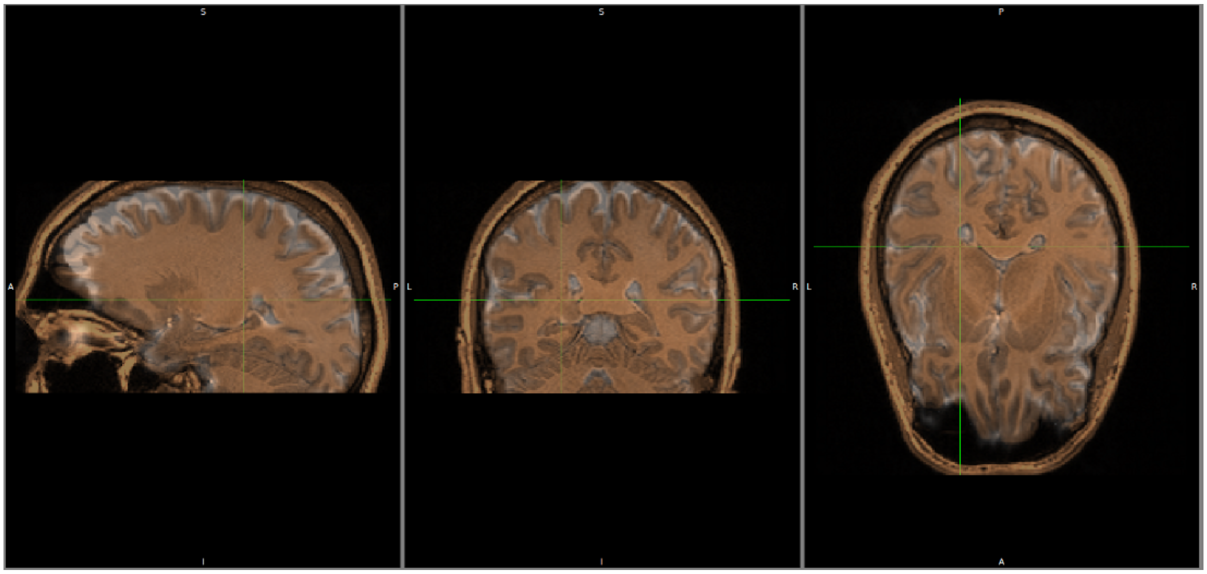
Registration

- Select the desired output from the list and click on `view`:



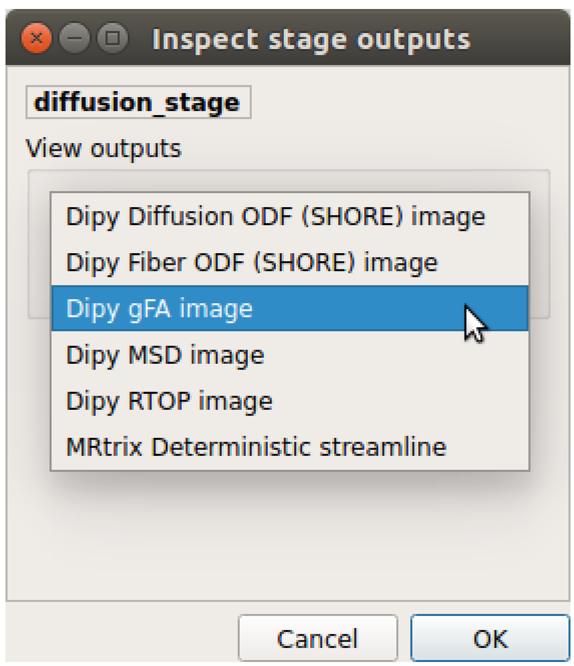
Registration results

Registration of T1 to Diffusion space (b0). T1 in copper overlayed to the b0 image.



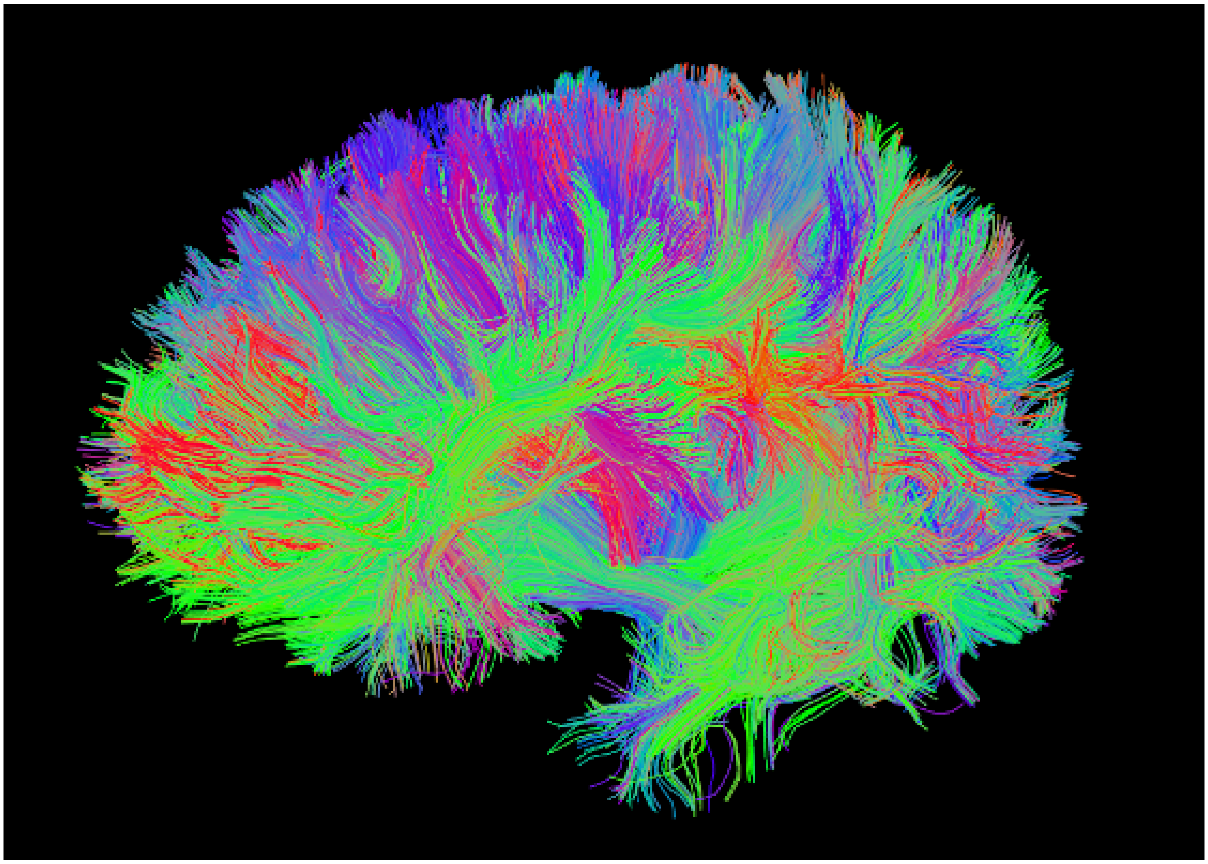
Diffusion reconstruction and tractography

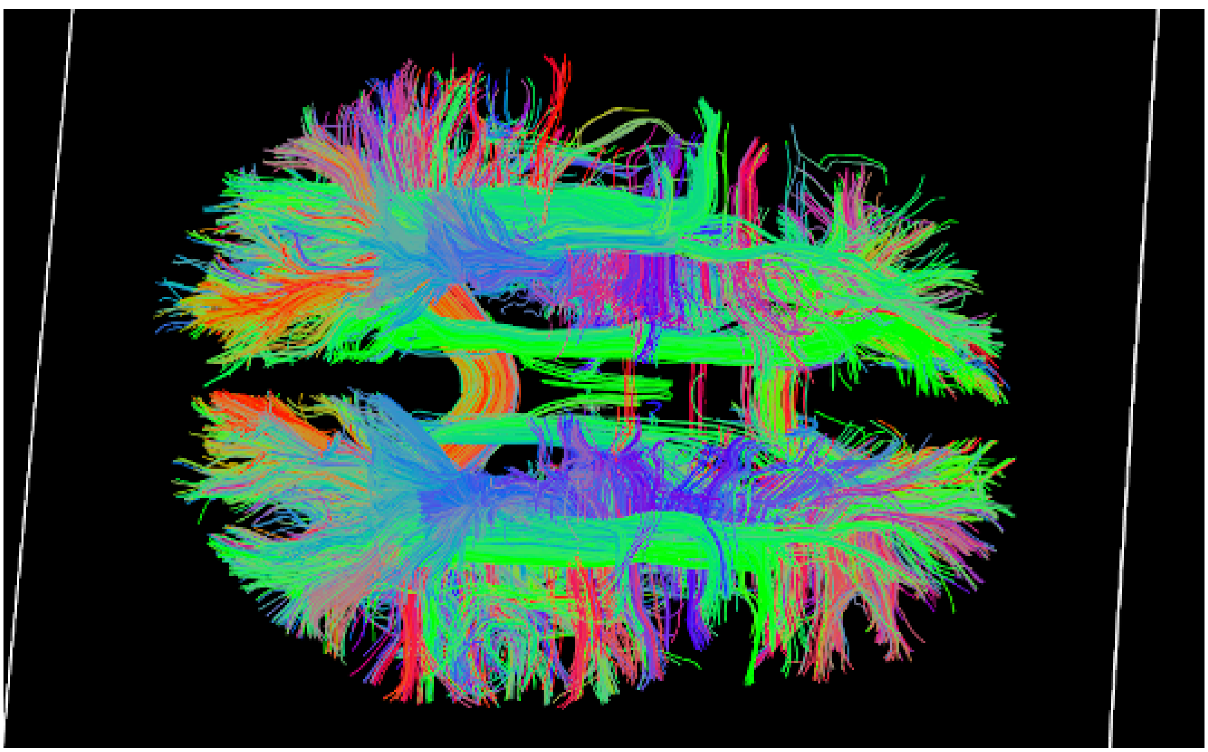
- Select the desired output from the list and click on `view`:



Tractography results

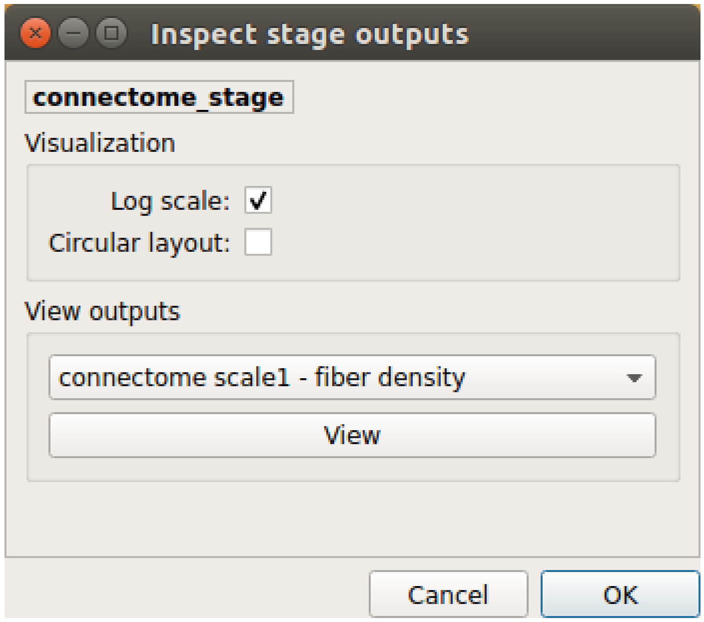
DSI Tractography results are displayed with TrackVis.





Connectome

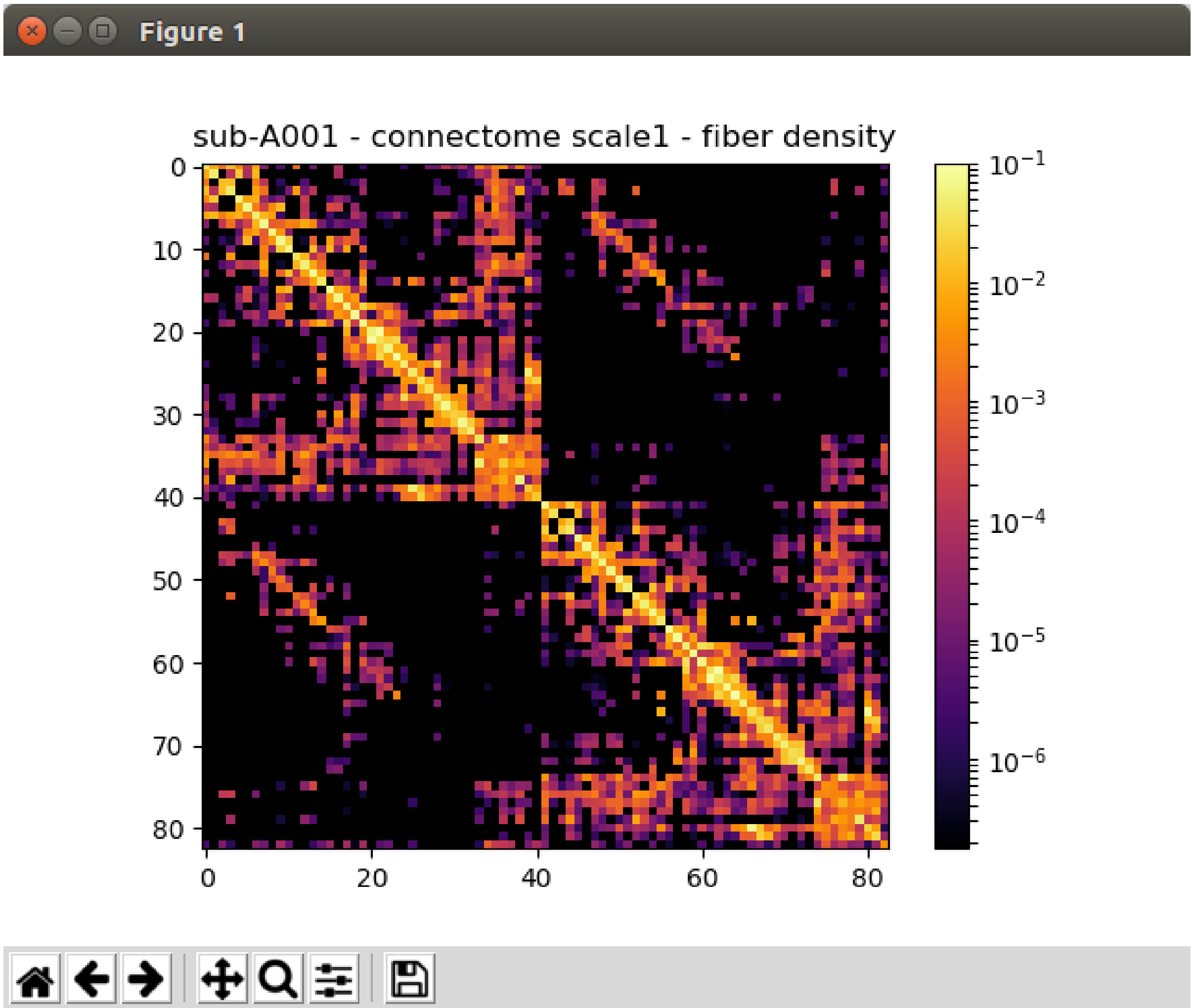
- Select the desired output from the list and click on `view`:



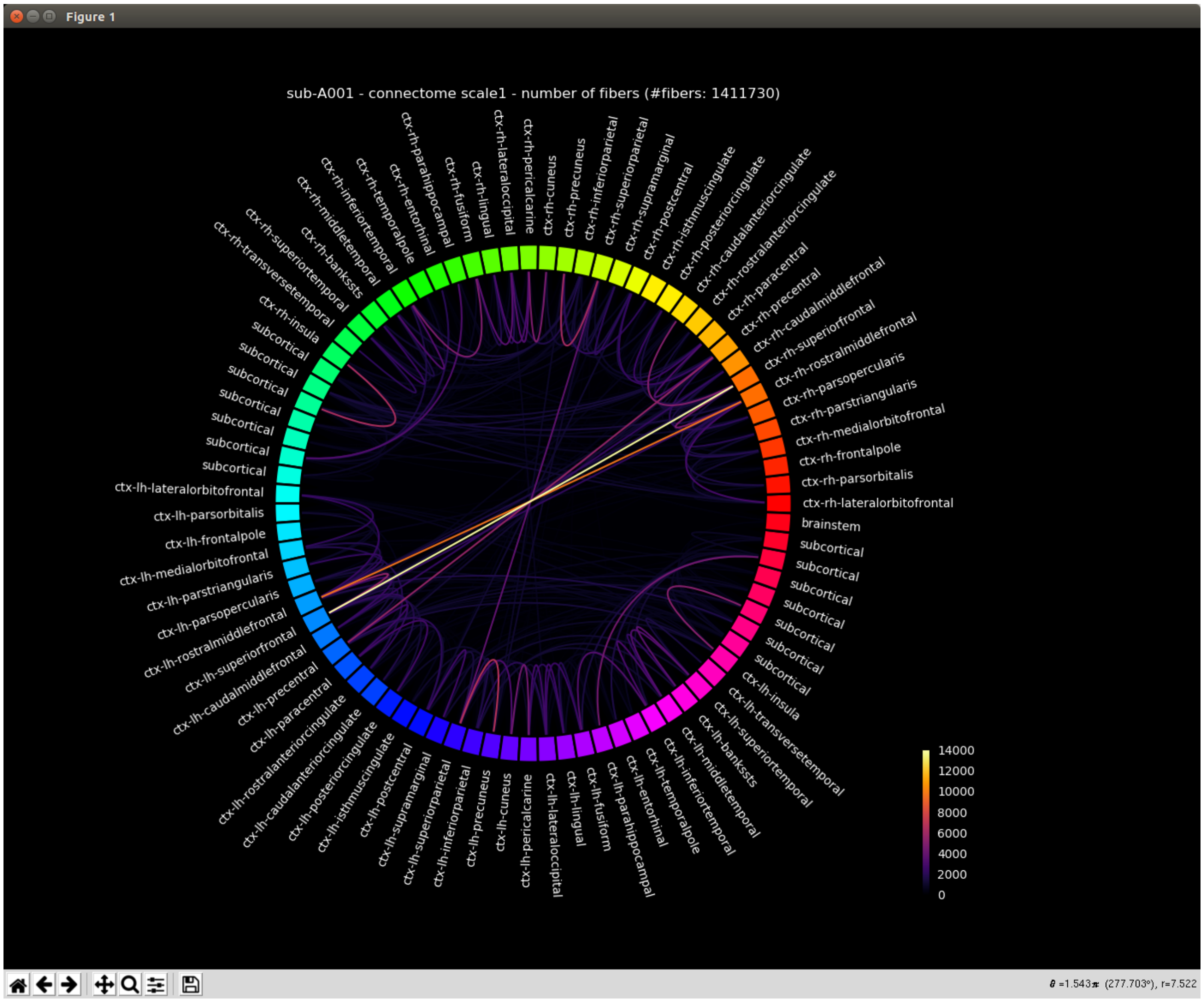
Generated connection matrix

Displayed using a:

1. matrix layout with pyplot



2. circular layout with pyplot and MNE



FMRI pipeline stages

- Click on the stage you wish to check the output(s):

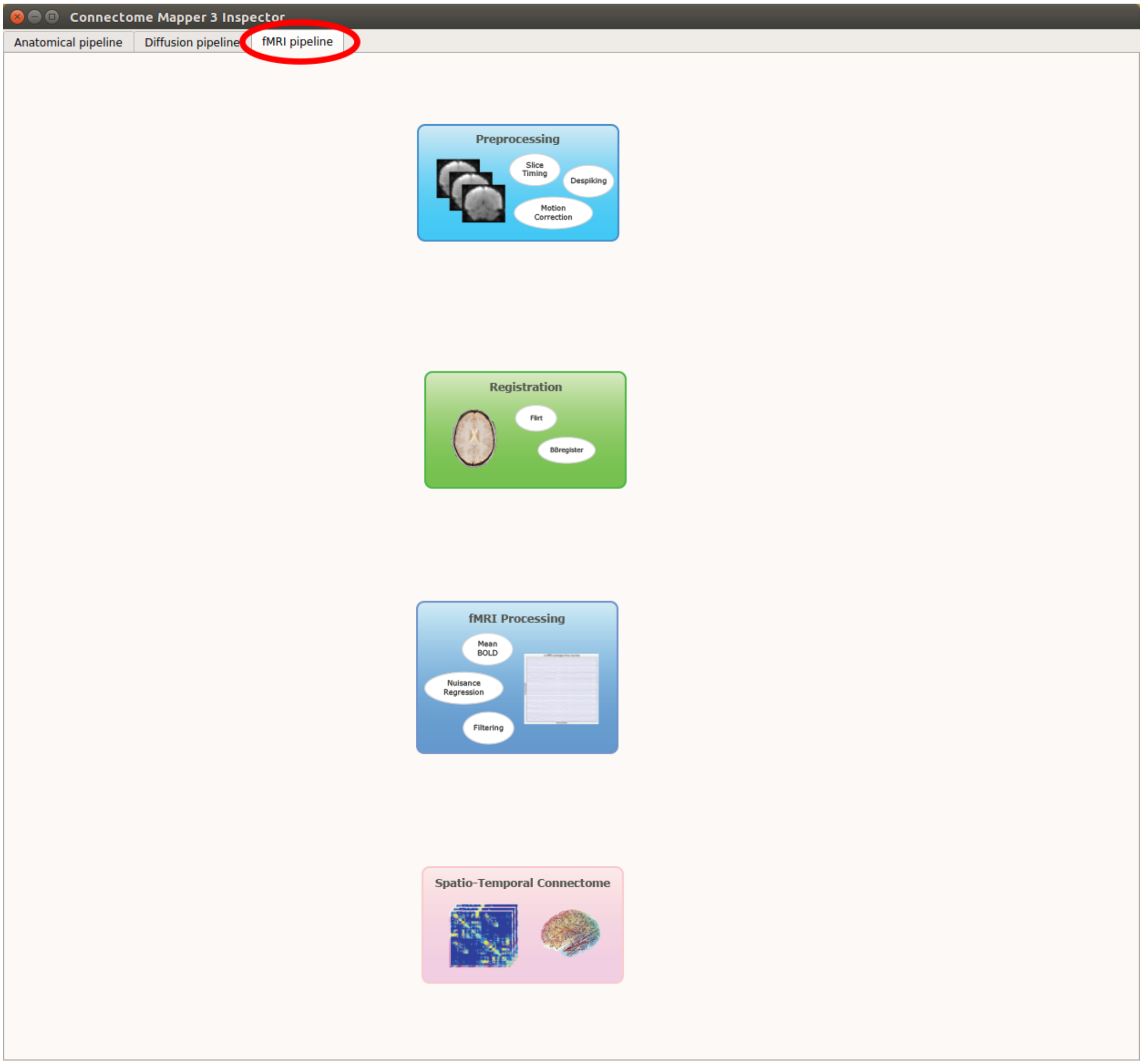
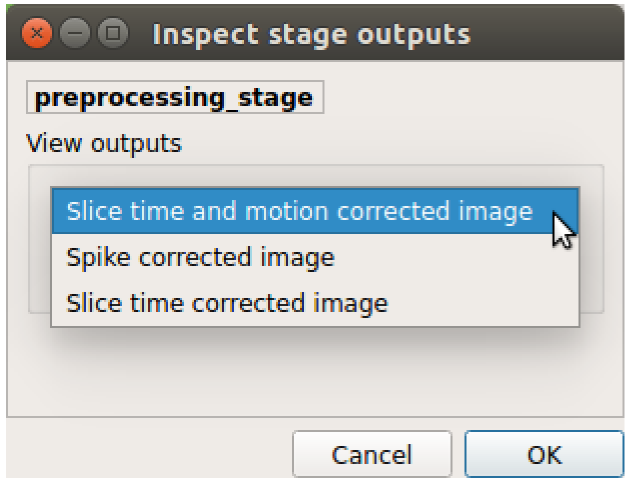


Fig. 11: Panel for output inspection of fMRI pipeline stages

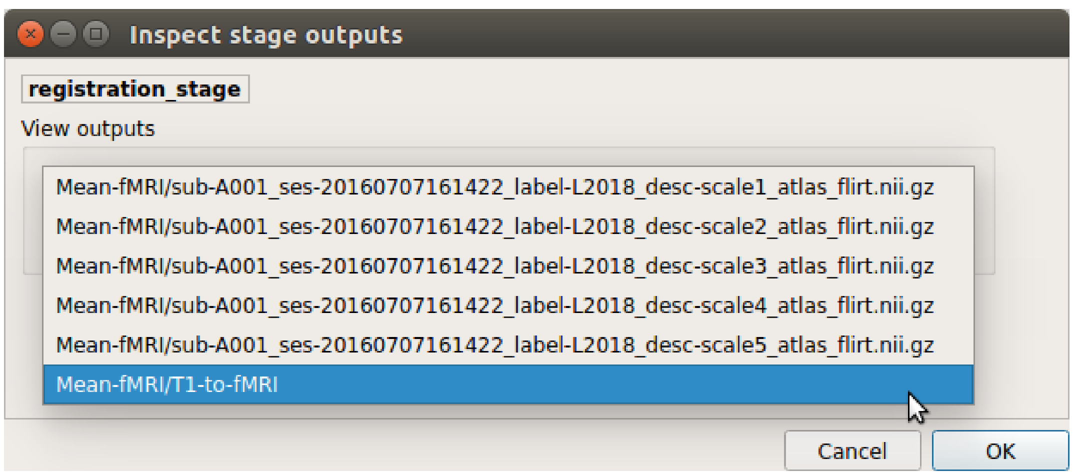
Preprocessing

- Select the desired output from the list and click on `view`:



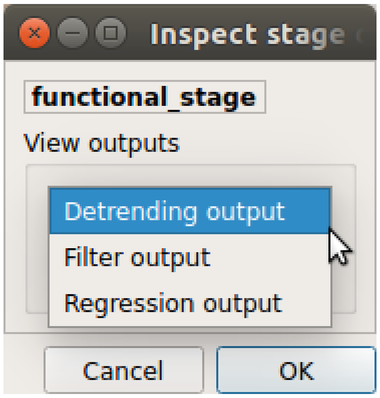
Registration

- Select the desired output from the list and click on `view`:

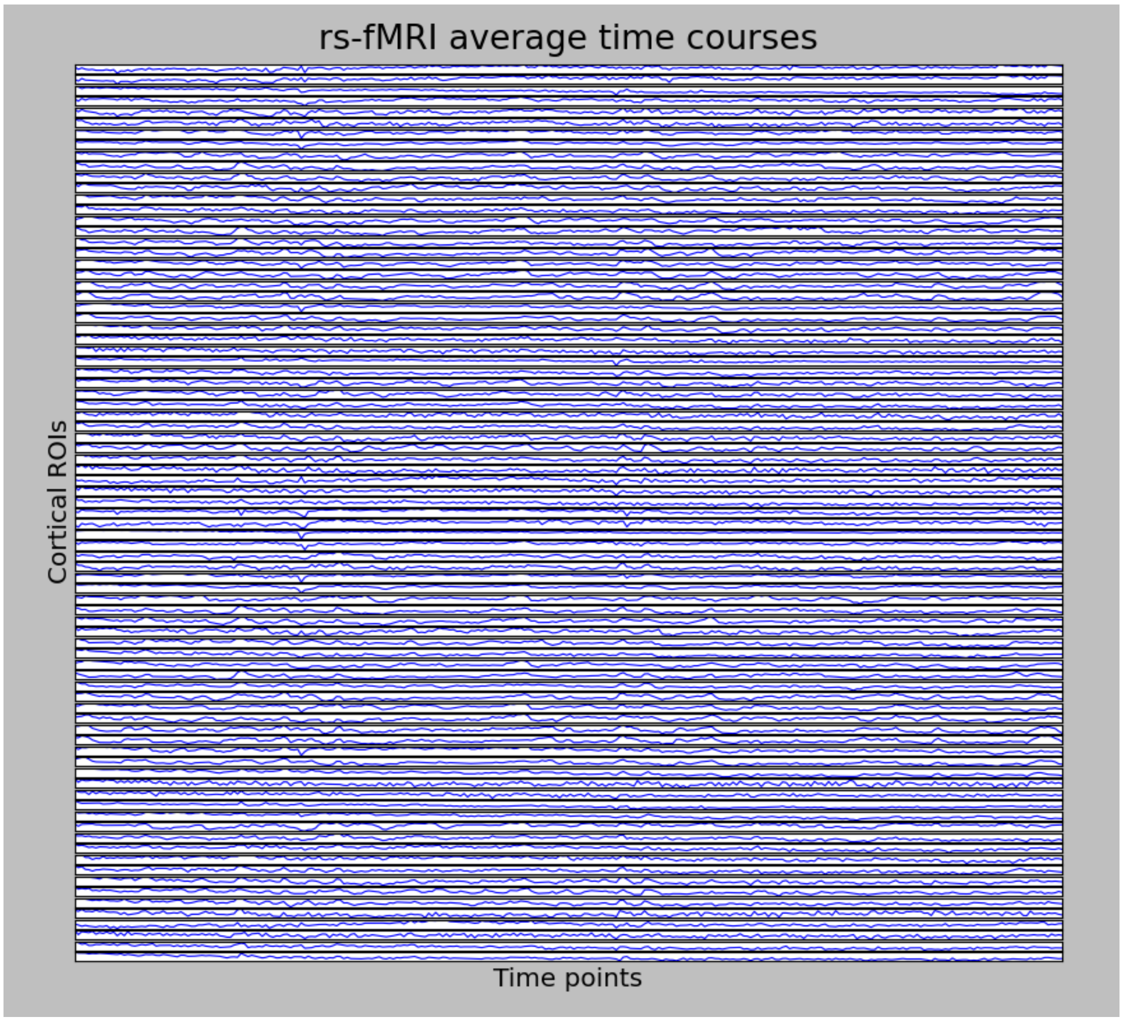


fMRI processing

- Select the desired output from the list and click on `view`:

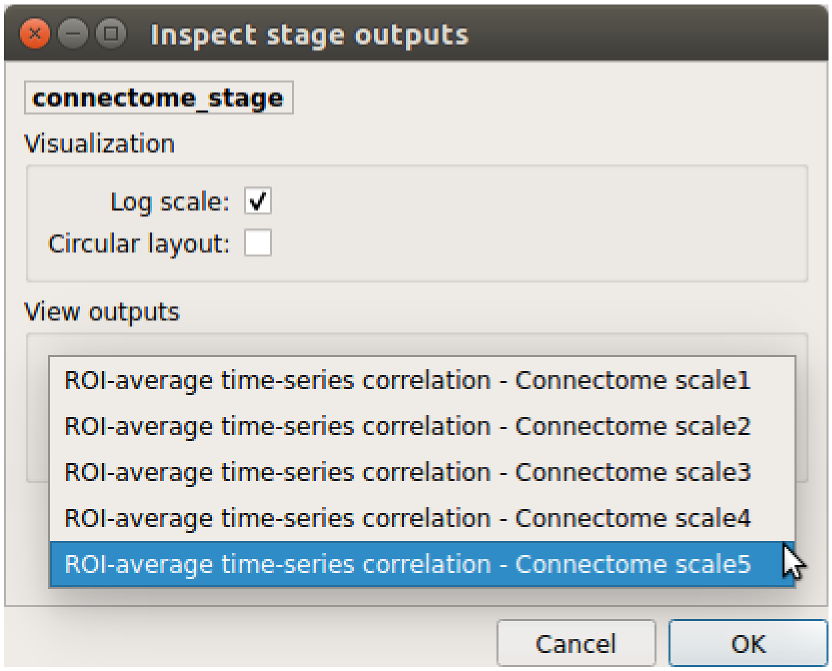


ROI averaged time-series



Connectome

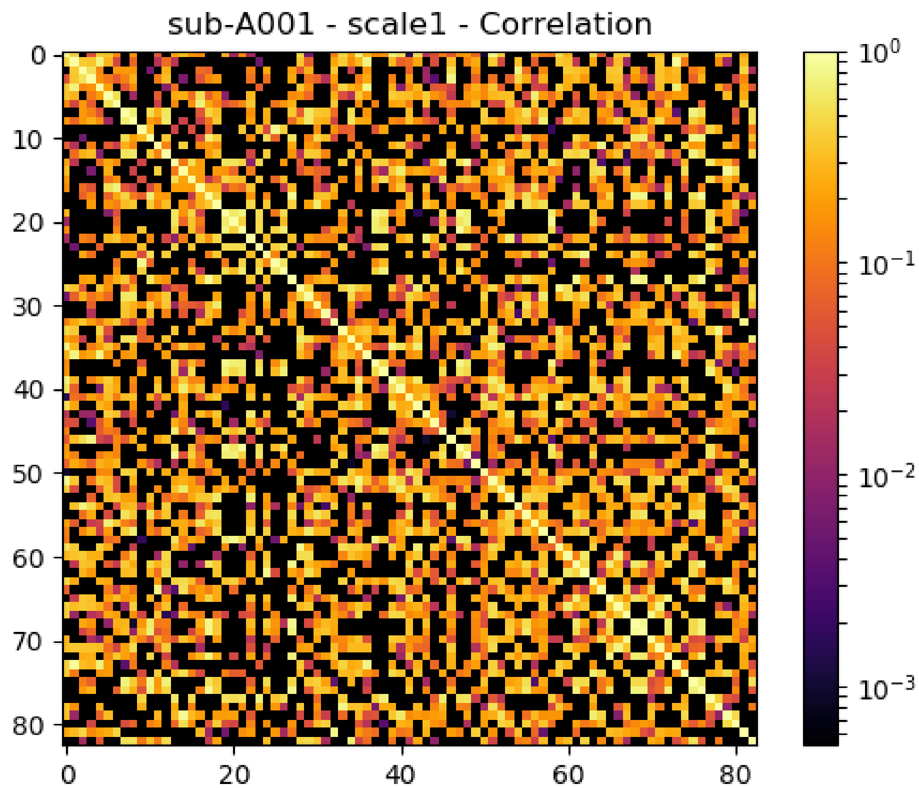
- Select the desired output from the list and click on `view`:



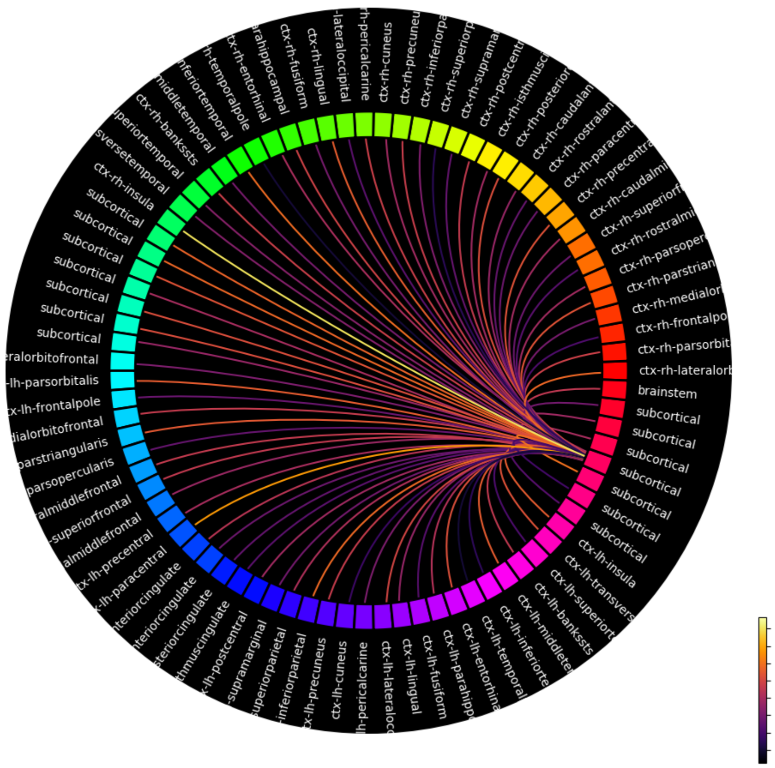
Generated connection matrix

Displayed using a:

1. matrix layout with pyplot



2. circular layout with pyplot and MNE



EEG pipeline stages

- Click on the stage you wish to check the output(s):

EEG Preprocessing

- Select the desired output from the list and click on `view`:

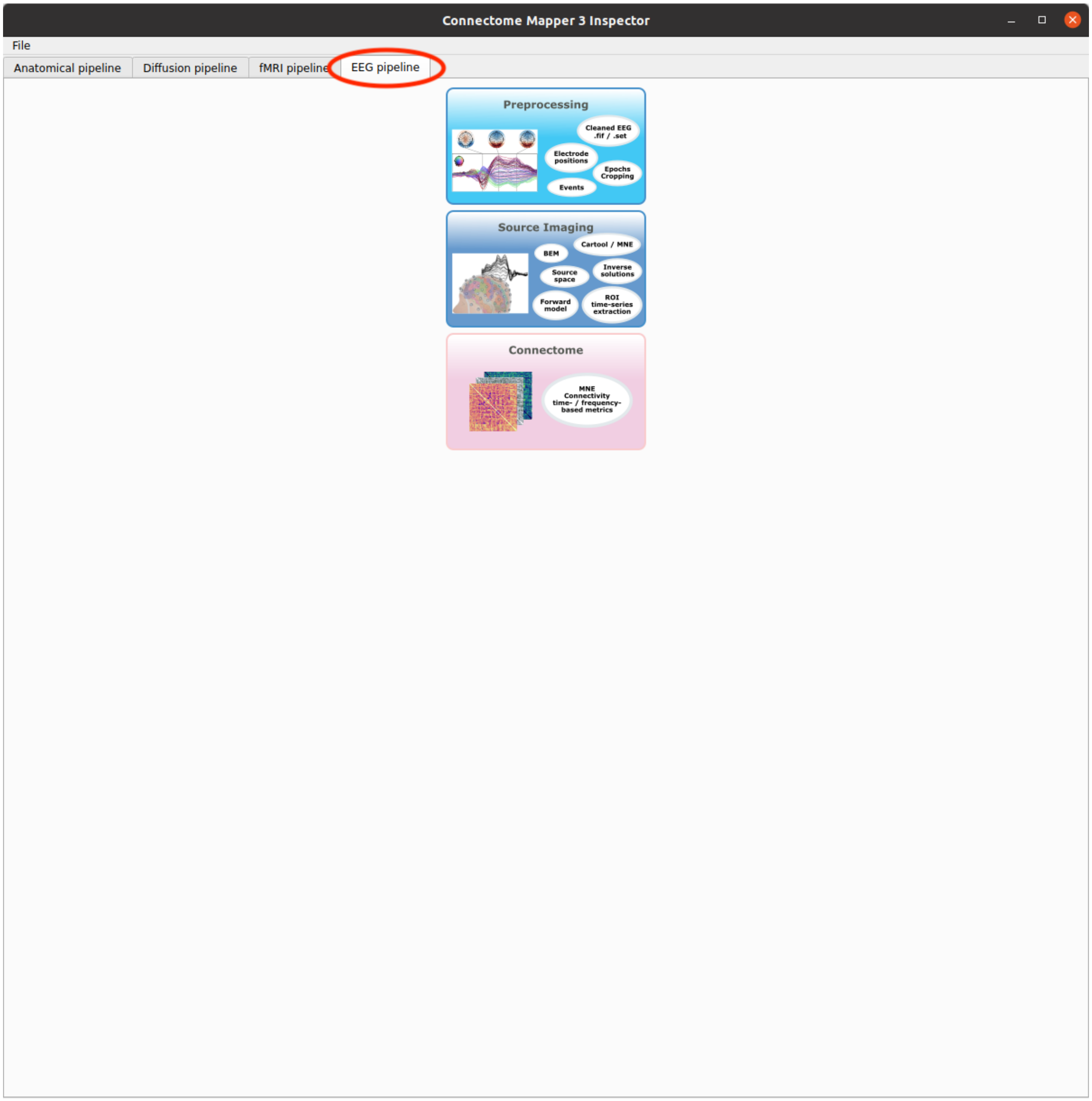
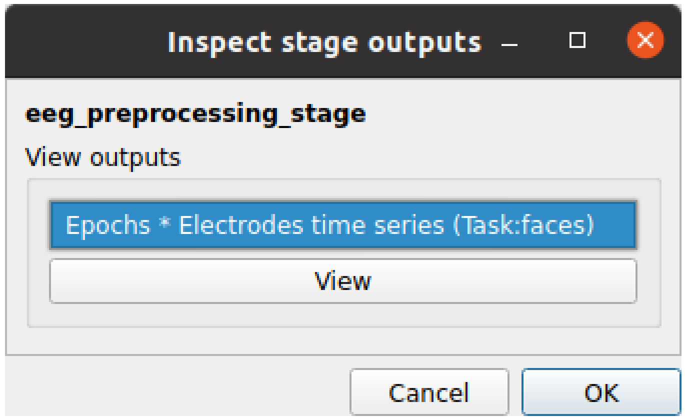
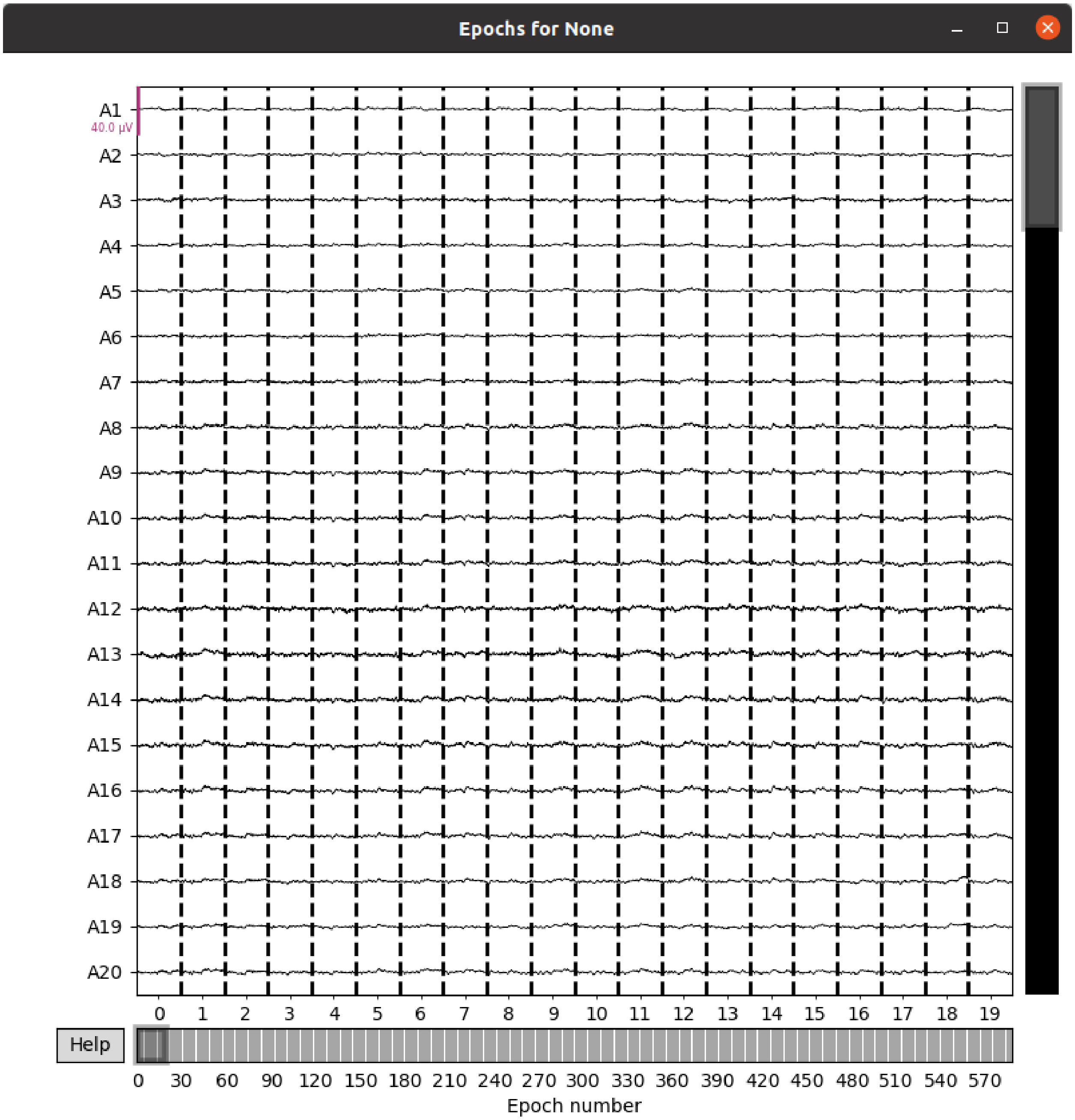


Fig. 12: Panel for output inspection of EEG pipeline stages



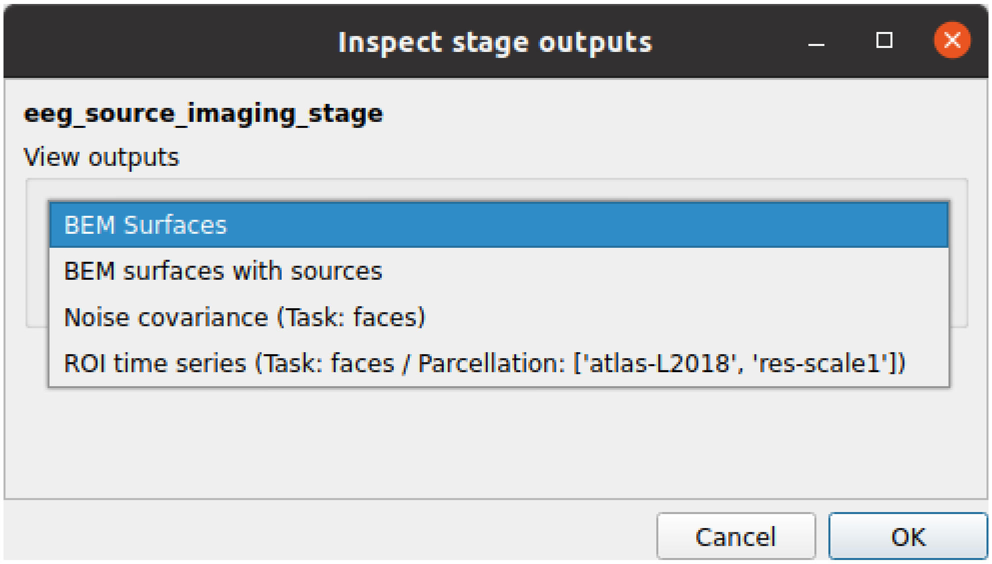
Epochs * Electrodes time-series

Plot saved `mne.Epochs` object.


Epochs for None
A1
40.0 µV
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
Help
0 30 60 90 120 150 180 210 240 270 300 330 360 390 420 450 480 510 540 570
Epoch number

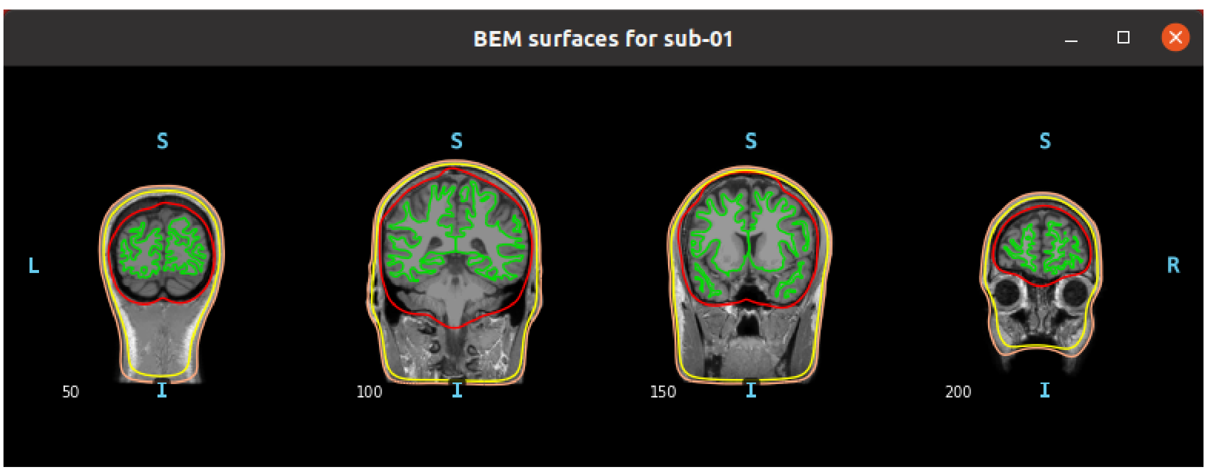
EEG Source Imaging

- Select the desired output from the list and click on `view`:



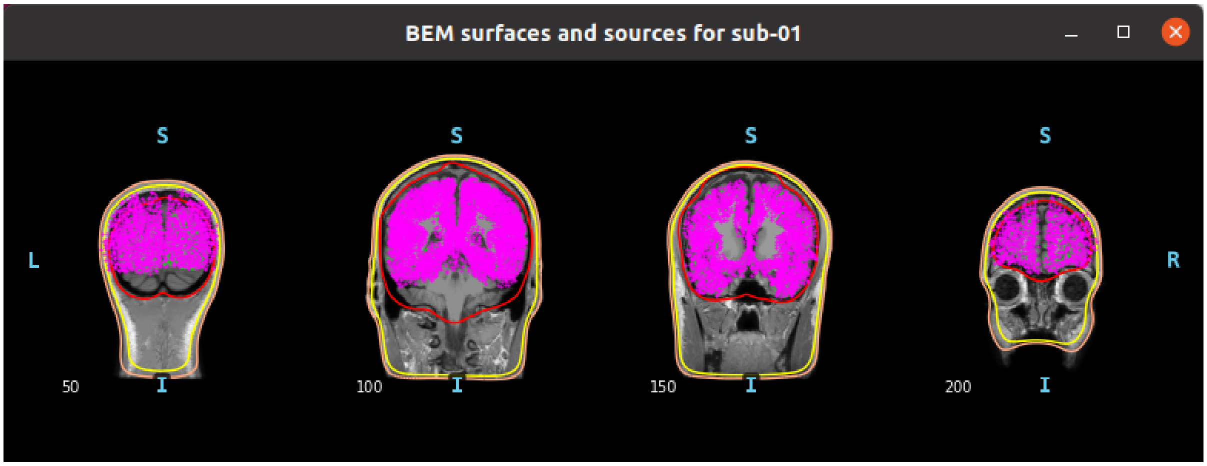
BEM surfaces

Surfaces of the boundary-element model used the MNE ESI workflow.



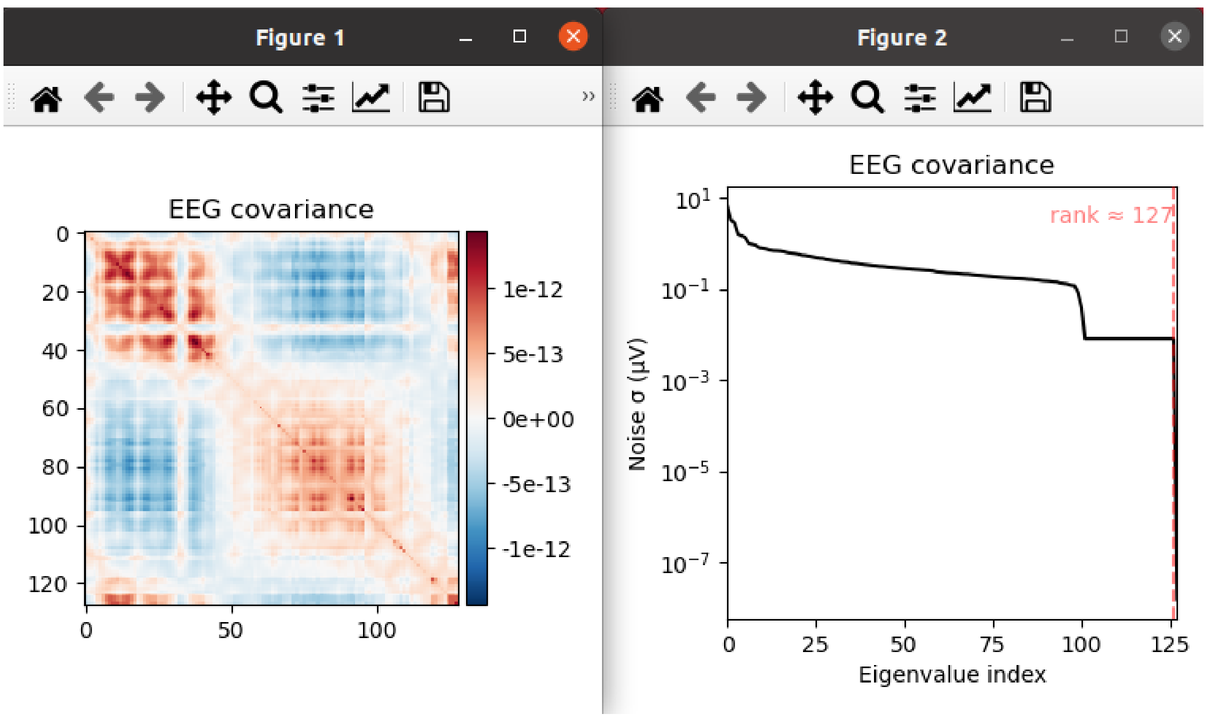
BEM surfaces with sources

Surfaces of the boundary-element model and sources used the MNE ESI workflow.



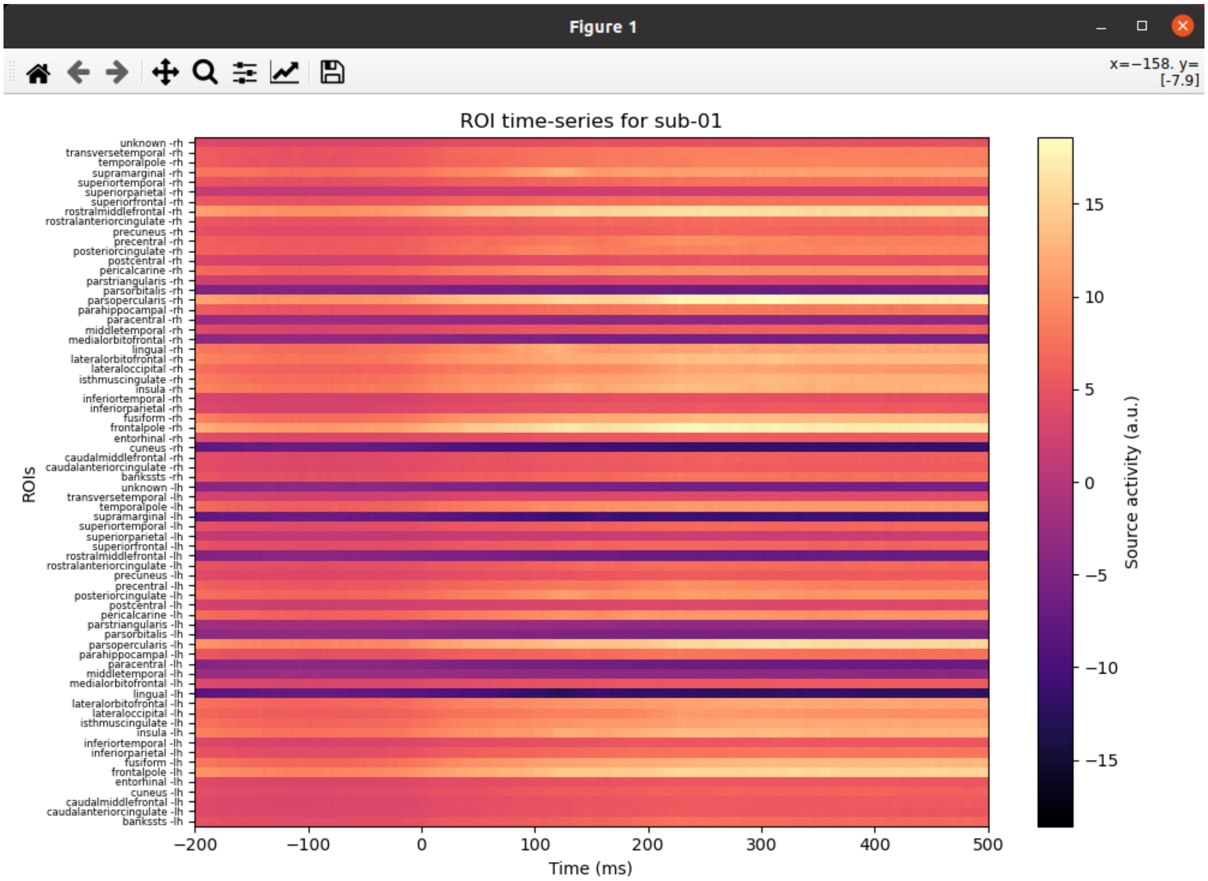
Noise covariance

Noise covariance matrix and spectrum estimated by the MNE ESI workflow.



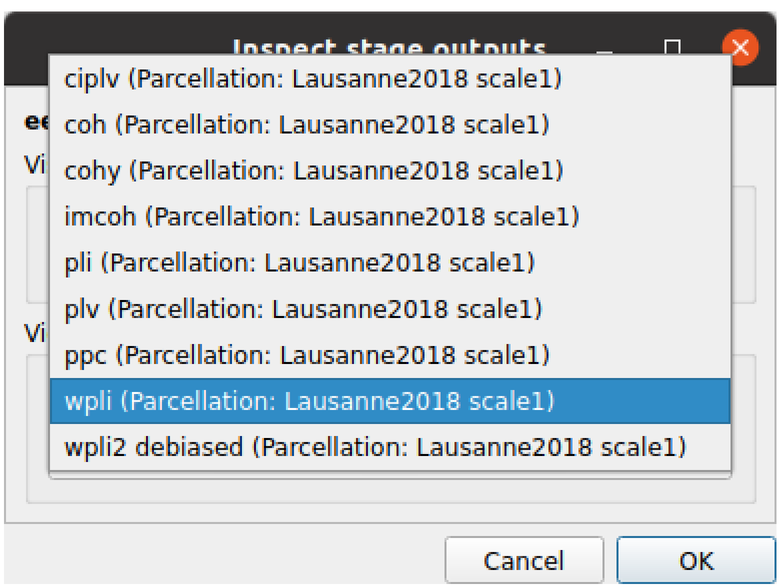
ROI time-series

Carpet plot of extracted ROI time-series.



EEG Connectome

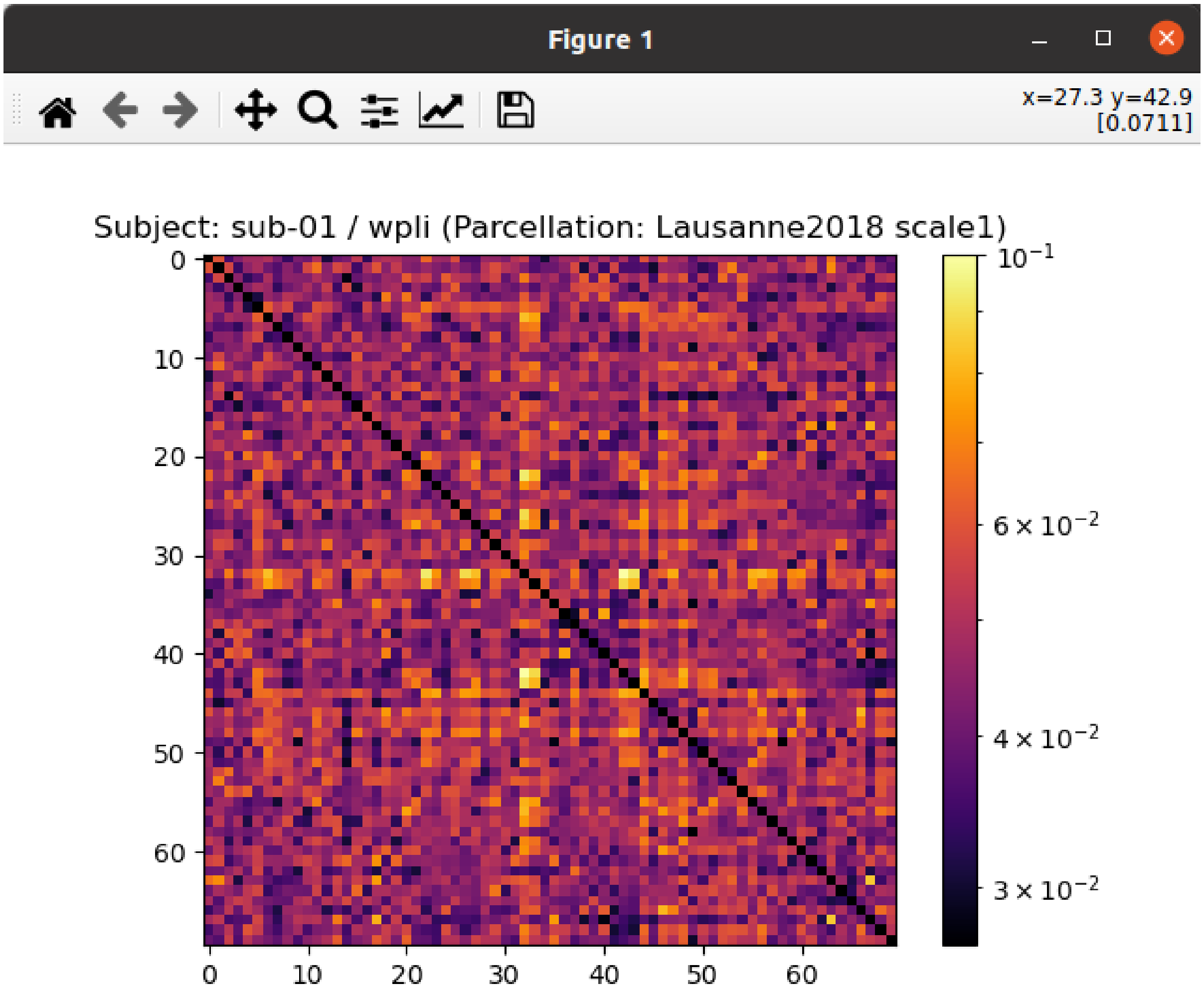
- Select the desired output from the list and click on `view`:



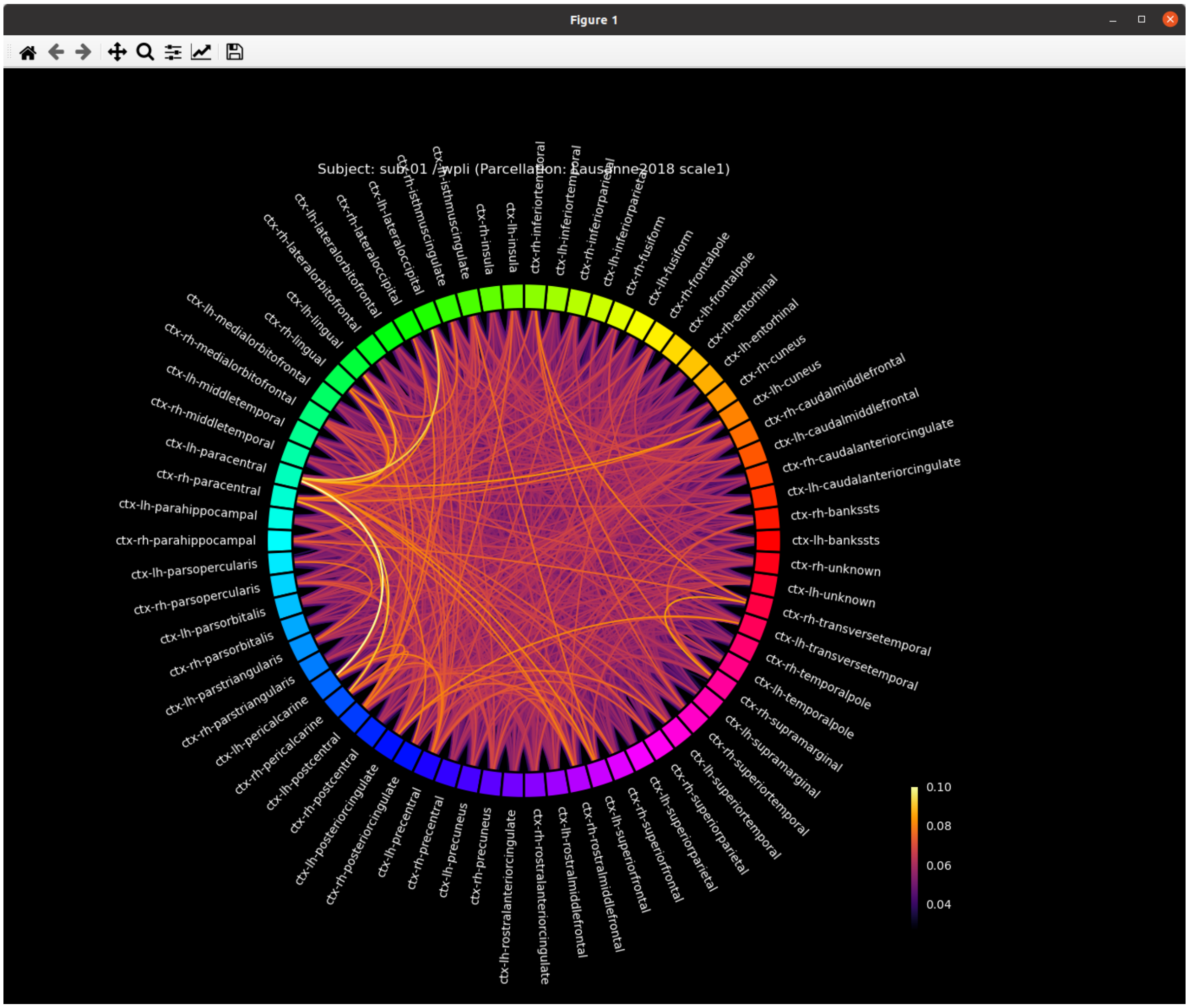
Generated connection matrix

Displayed using a:

1. matrix layout with pyplot



2. circular layout with pyplot and MNE



5.5 Outputs of Connectome Mapper 3

Processed, or derivative, data are outputed to <bids_dataset/derivatives>/.

5.5.1 BIDS derivatives entities

Entity	Description
sub-<label>	Distinguish different subjects
ses-<label>	Distinguish different acquisition sessions
task-<label>	Distinguish different experiment tasks
label-<label>	Describe the type of brain tissue segmented (for _probseg/dseg)
atlas-<label>	Distinguish data derived from different types of parcellation atlases
res-<label>	Distinguish data derived from the different scales of Lausanne2008 and Lausanne2018 parcellation atlases
space-DWI	Distinguish anatomical MRI derivatives in the target diffusion MRI space
model-<label>	Distinguish different diffusion signal models (DTI, CSD, SHORE, MAPMRI)

See Original BIDS Entities Appendix for more description.

Note: Connectome Mapper 3 introduced a new BIDS entity `atlas-<atlas_label>` (where `<atlas_label>`: `Desikan`/ `L2018`), that is used in combination with the `res-<atlas_scale>` (where `<atlas_scale>`: `scale1` / `scale2` / `scale3` / `scale4` / `scale5`) entity to distinguish data derived from different parcellation atlases and different scales.

5.5.2 Main Connectome Mapper Derivatives

Main outputs produced by Connectome Mapper 3 are written to `cmp/sub-<subject_label>/`. In this folder, a configuration file generated for each modality pipeline (i.e. anatomical/diffusion/fMRI/EEG) and used for processing each participant is saved as `sub-<subject_label>_anatomical/diffusion/fMRI/EEG_config.json`. It summarizes pipeline workflow options and parameters used for processing. An execution log of the full workflow is saved as `sub-<subject_label>_log.txt``.

Anatomical derivatives

- Anatomical derivatives in the individual `T1w` space are placed in each subject's `anat/` subfolder, including:
  - The original T1w image:
    - `anat/sub-<subject_label>_desc-head_T1w.nii.gz`
  - The masked T1w image with its corresponding brain mask:
    - `anat/sub-<subject_label>_desc-brain_T1w.nii.gz`
    - `anat/sub-<subject_label>_desc-brain_mask.nii.gz`
  - The segmentations of the white matter (WM), gray matter (GM), and Cortical Spinal Fluid (CSF) tissues:

    * anat/sub-<subject_label>_label-WM_dseg.nii.gz
    * anat/sub-<subject_label>_label-GM_dseg.nii.gz
    * anat/sub-<subject_label>_label-CSF_dseg.nii.gz
  - The five different brain parcellations:
    * anat/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_dseg.nii.gz

      where:
      - <atlas_label>: Desikan / L2018 is the parcellation scheme used
      - <scale_label>: scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale if applicable

    with two tsv side-car files that follow the BIDS derivatives, one describing the parcel label/index mapping (_dseg.tsv), one reporting volumetry of the different parcels (_stats.tsv), and two files used internally by CMP3, one describing the parcel labels in graphml format (dseg.graphml), one providing the color lookup table of the parcel labels in Freesurfer format which can used directly in freeview (_FreeSurferColorLUT.txt):

    * anat/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_dseg.tsv
    * anat/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_stats.tsv
    * anat/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_dseg.graphml
    * anat/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_FreeSurferColorLUT.txt

- Anatomical derivatives in the DWI space produced by the diffusion pipeline are placed in each subject's anat/ subfolder, including:
  - The unmasked T1w image:
    * anat/sub-<subject_label>_space-DWI_desc-head_T1w.nii.gz
  - The masked T1w image with its corresponding brain mask:
    * anat/sub-<subject_label>_space-DWI_desc-brain_T1w.nii.gz
    * anat/sub-<subject_label>_space-DWI_desc-brain_mask.nii.gz
  - The segmentation of WM tissue used for tractography seeding:
    * anat/sub-<subject_label>_space-DWI_label-WM_dseg.nii.gz
  - The five different brain parcellation are saved as:
    * anat/sub-<subject_label>_space-DWI_atlas-<atlas_label>[_res-<scale_label>]_dseg.nii.gz

      where:
      - <atlas_label>: Desikan / L2018 is the parcellation scheme used
      - <scale_label>: scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale if applicable
  - The 5TT image used for Anatomically Constrained Tractorgaphy (ACT):

    * anat/sub-<subject_label>_space-DWI_label-5TT_probseg.nii.gz

  - The patial volume maps for white matter (WM), gray matter (GM), and Cortical Spinal Fluid (CSF) used for Particale Filtering Tractography (PFT), generated from 5TT image:
    * anat/sub-<subject_label>_space-DWI_label-WM_probseg.nii.gz
    * anat/sub-<subject_label_space-DWI>_label-GM_probseg.nii.gz
    * anat/sub-<subject_label>_space-DWI_label-CSF_probseg.nii.gz
  - The GM/WM interface used for ACT and PFT seeding:
    * anat/sub-<subject_label>_space-DWI_label-GMWMI_probseg.nii.gz

Diffusion derivatives

Diffusion derivatives in the individual DWI space are placed in each subject's dwi/ subfolder, including:

- The final preprocessed DWI image used to fit the diffusion model for tensor or fiber orientation distribution estimation:
  - dwi/sub-<subject_label>_desc-preproc_dwi.nii.gz
- The brain mask used to mask the DWI image:
  - dwi/sub-<subject_label>_desc-brain_mask_resampled.nii.gz
- The diffusion tensor (DTI) fit (if used for tractography):
  - dwi/sub-<subject_label>]_desc-WLS_model-DTI_diffmodel.nii.gz

    with derived Fractional Anisotropic (FA) and Mean Diffusivity (MD) maps:
  - dwi/sub-<subject_label>]_model-DTI_FA.nii.gz
  - dwi/sub-<subject_label>]_model-DTI_MD.nii.gz
- The Fiber Orientation Distribution (FOD) image from Constrained Spherical Deconvolution (CSD) fit (if performed):
  - dwi/sub-<subject_label>]_model-CSD_diffmodel.nii.gz
- The MAP-MRI fit for DSI and multi-shell DWI data (if performed):
    - dwi/sub-<subject_label>]_model-MAPMRI_diffmodel.nii.gz

  with derived Generalized Fractional Anisotropic (GFA), Mean Squared Displacement (MSD), Return-to-Origin Probability (RTOP) and Return-to-Plane Probability (RTPP) maps:
    - dwi/sub-<subject_label>]_model-MAPMRI_GFA.nii.gz
    - dwi/sub-<subject_label>]_model-MAPMRI_MSD.nii.gz
    - dwi/sub-<subject_label>]_model-MAPMRI_RTOP.nii.gz
    - dwi/sub-<subject_label>]_model-MAPMRI_RTPP.nii.gz
- The SHORE fit for DSI data:
    - dwi/sub-<subject_label>]_model-SHORE_diffmodel.nii.gz

  with derived Generalized Fractional Anisotropic (GFA), Mean Squared Displacement (MSD), Return-to-Origin Probability (RTOP) maps:
    - dwi/sub-<subject_label>]_model-SHORE_GFA.nii.gz
    - dwi/sub-<subject_label>]_model-SHORE_MSD.nii.gz

    - dwi/sub-<subject_label>]_model-SHORE_RTOP.nii.gz

- The tractogram:
  - dwi/sub-<subject_label>_model-<model_label>_desc-<label>_tractogram.trk

    where:

    * <model_label> is the diffusion model used to drive tractography (DTI, CSD, SHORE)
    * <label> is the type of tractography algorithm employed (DET for deterministic, PROB for probabilistic)

- The structural connectivity (SC) graphs:
  - dwi/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_conndata-network_connectivity.<fmt>

    where:

    * <atlas_label>: Desikan / L2018 is the parcellation scheme used
    * <scale_label>: scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale if applicable
    * <fmt>: mat / gpickle / tsv / graphml is the format used to store the graph

Functional derivatives

Functional derivatives in the 'meanBOLD' (individual) space are placed in each subject's func/ subfolder including:

- The original BOLD image:
  - func/sub-<subject_label>_task-rest_desc-cmp_bold.nii.gz
- The mean BOLD image:
  - func/sub-<subject_label>_meanBOLD.nii.gz
- The fully preprocessed band-pass filtered used to compute ROI time-series:
  - func/sub-<subject_label>_desc-bandpass_task-rest_bold.nii.gz
- For scrubbing (if enabled):
  - The change of variance (DVARS):
    * func/sub-<subject_label>_desc-scrubbing_DVARS.npy
  - The frame displacement (FD):
    * func/sub-<subject_label>_desc-scrubbing_FD.npy
- Motion-related time-series:
  - func/sub-<subject_label>_motion.tsv
- The ROI time-series for each parcellation scale:
  - func/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_timeseries.npy
  - func/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_timeseries.mat

    where:

    * <atlas_label>: Desikan / L2018 is the parcellation scheme used

    * `<scale_label>`: `scale1`, `scale2`, `scale3`, `scale4`, `scale5` corresponds to the parcellation scale if applicable

- The functional connectivity (FC) graphs:
  - `func/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_conndata-network_connectivity.<fmt>`

    where:

    * `<atlas_label>`: `Desikan` / `L2018` is the parcellation scheme used
    * `<scale_label>`: `scale1`, `scale2`, `scale3`, `scale4`, `scale5` corresponds to the parcellation scale if applicable
    * `<fmt>`: `mat` / `gpickle` / `tsv` / `graphml` is the format used to store the graph

EEG derivatives

EEG derivatives are placed in each subject's `eeg/` subfolder including:

- The preprocessed EEG epochs data in `fif` format:
  - `eeg/sub-<subject_label>_task-<task_label>_epo.fif`
- The BEM surfaces in `fif` format:
  - `eeg/sub-<subject_label>_task-<task_label>_bem.fif`
- The source space in `fif` format:
  - `eeg/sub-<subject_label>_task-<task_label>_src.fif`
- The forward solution in `fif` format:
  - `eeg/sub-<subject_label>_task-<task_label>_fwd.fif`
- The inverse operator in `fif` format:
  - `eeg/sub-<subject_label>_task-<task_label>_inv.fif`
- The computed noise covariance in `fif` format:
  - `eeg/sub-<subject_label>_task-<task_label>_noisecov.fif`
- The transform of electrode positions that might be used for ESI in `fif` format:
  - `eeg/sub-<subject_label>_trans.fif`
- The ROI time-series for each parcellation atlas (and scale):
  - `eeg/sub-<subject_label>_task-<task_label>_atlas-<atlas_label>[_res-<scale_label>]_timeseries.npy`
  - `eeg/sub-<subject_label>_task-<task_label>_atlas-<atlas_label>[_res-<scale_label>]_timeseries.mat`

    where:

    * `<atlas_label>`: `Desikan` / `L2018` is the parcellation scheme used
    * `<scale_label>`: `scale1`, `scale2`, `scale3`, `scale4`, `scale5` corresponds to the parcellation scale if applicable

- The functional frequency- and time-frequency-domain based connectivity graphs:

– eeg/sub-<subject_label>_task-<task_label>_atlas-<atlas_label>[_res-<scale_label>]_conndata-netw
<fmt>

where:

* <atlas_label>: Desikan / L2018 is the parcellation scheme used
* <scale_label>: scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale if applicable
* <fmt>: mat / gpickle / tsv / graphml is the format used to store the graph

5.5.3 FreeSurfer Derivatives

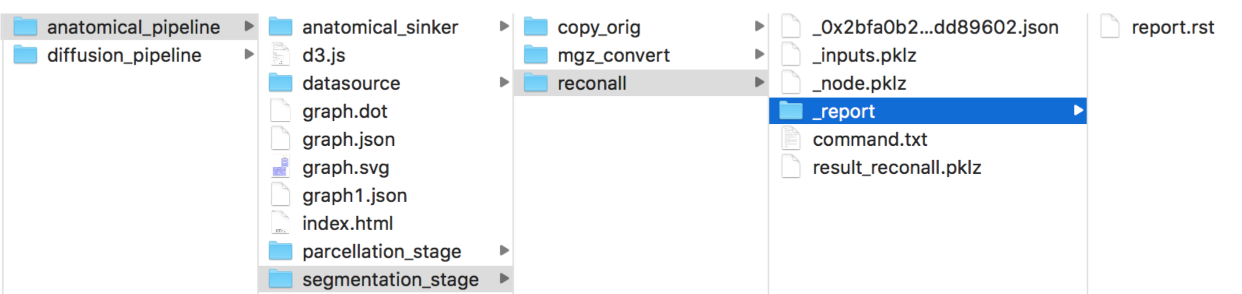
A FreeSurfer subjects directory is created in <bids_dataset/derivatives>/freesurfer-7.2.0.

```
freesurfer-7.1.1/
    fsaverage/
        mri/
        surf/
        ...
    sub-<subject_label>/
        mri/
        surf/
        ...
    ...
```

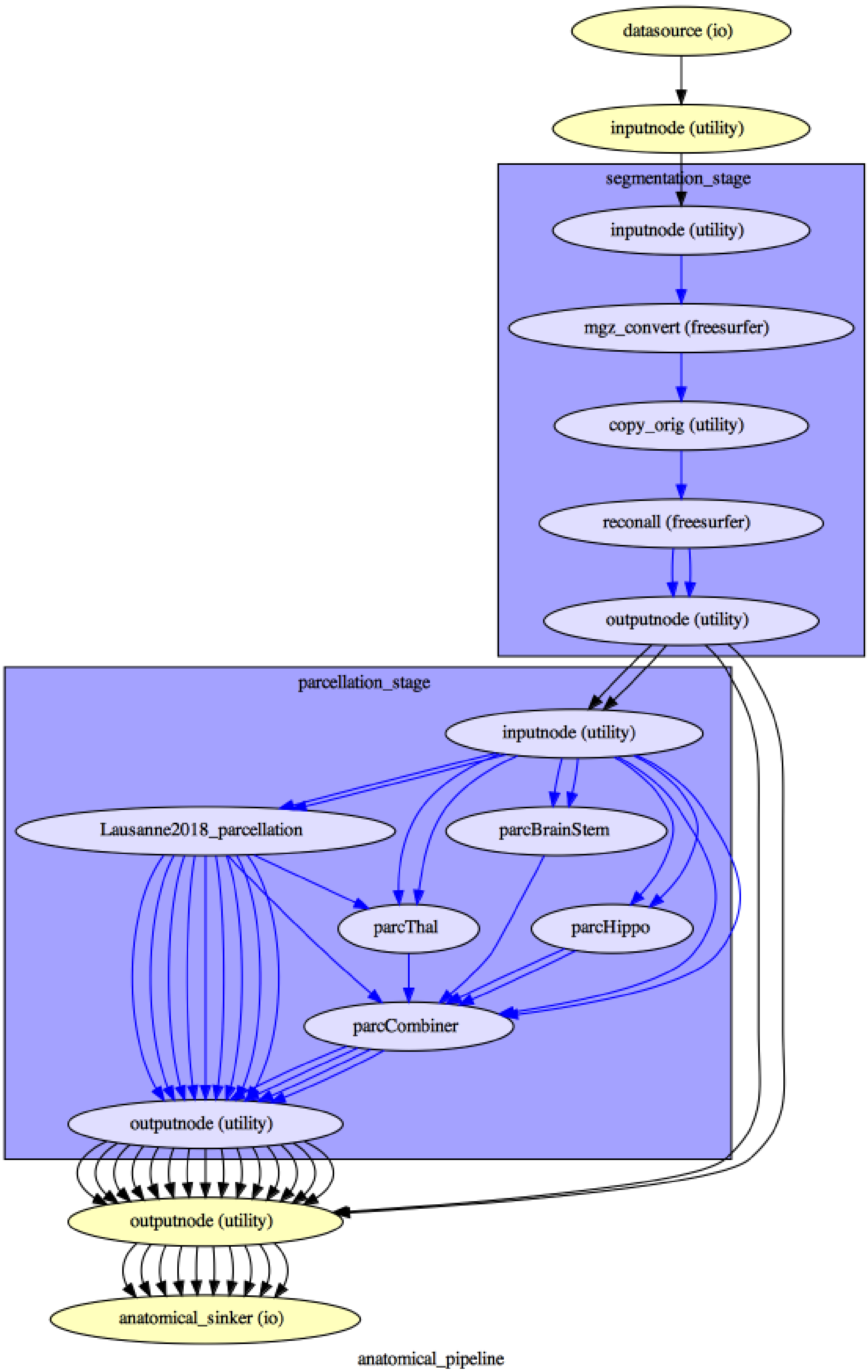
The fsaverage subject distributed with the running version of FreeSurfer is copied into this directory.

5.5.4 Nipype Workflow Derivatives

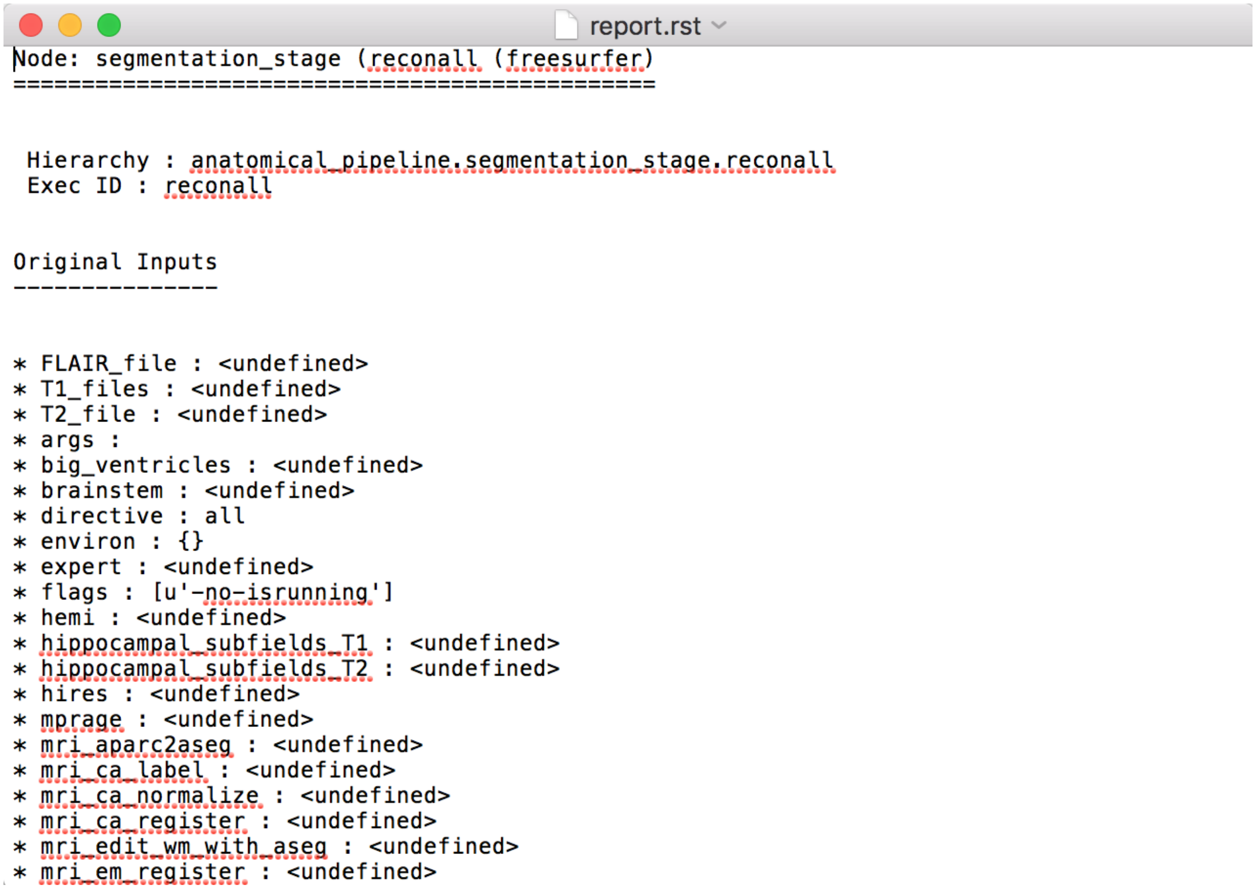
The execution of each Nipype workflow (pipeline) dedicated to the processing of one modality (i.e. anatomical/diffusion/fMRI/EEG) involves the creation of a number of intermediate outputs which are written to <bids_dataset/derivatives>/nipype/sub-<subject_label>/<anatomical/diffusion/fMRI/eeg>_pipeline respectively:



To enhance transparency on how data is processed, outputs include a pipeline execution graph saved as <anatomical/diffusion/fMRI/eeg>_pipeline/graph.svg which summarizes all processing nodes involves in the given processing pipeline:


datasource (io)
inputnode (utility)
segmentation_stage
inputnode (utility)
mgz_convert (freesurfer)
copy_orig (utility)
reconall (freesurfer)
outputnode (utility)
parcellation_stage
inputnode (utility)
Lausanne2018_parcellation
parcBrainStem
parcThal
parcHippo
parcCombiner
outputnode (utility)
outputnode (utility)
anatomical_sinker (io)
anatomical_pipeline

Execution details (data provenance) of each interface (node) of a given pipeline are reported in <anatomical/diffusion/fMRI/eeg>_pipeline/<stage_name>/<interface_name>/_report/report.rst



report.rst

```
Node: segmentation_stage (reconall (freesurfer)
===============================================


 Hierarchy : anatomical_pipeline.segmentation_stage.reconall
 Exec ID : reconall


Original Inputs
---------------


* FLAIR_file : <undefined>
* T1_files : <undefined>
* T2_file : <undefined>
* args :
* big_ventricles : <undefined>
* brainstem : <undefined>
* directive : all
* environ : {}
* expert : <undefined>
* flags : [u'-no-isrunning']
* hemi : <undefined>
* hippocampal_subfields_T1 : <undefined>
* hippocampal_subfields_T2 : <undefined>
* hires : <undefined>
* mprage : <undefined>
* mri_aparc2aseg : <undefined>
* mri_ca_label : <undefined>
* mri_ca_normalize : <undefined>
* mri_ca_register : <undefined>
* mri_edit_wm_with_aseg : <undefined>
* mri_em_register : <undefined>
```

Note: Connectome Mapper 3 outputs are currently being updated to conform to BIDS v1.4.0.

5.6 Packages and modules

5.6.1 cmp package

Submodules

cmp.parser module

Connectome Mapper 3 Commandline Parser.

`cmp.parser.`**`get`**`()` → argparse.ArgumentParser

Return the argparse parser of the BIDS App.

Returns **p** – Instance of `argparse.ArgumentParser`

Return type argparse.ArgumentParser

`cmp.parser.`**`get_docker_wrapper_parser`**`()` → argparse.ArgumentParser

Return the argparse parser of the Docker BIDS App.

Returns **p** – Instance of `argparse.ArgumentParser`

Return type argparse.ArgumentParser

`cmp.parser.get_singularity_wrapper_parser()` → argparse.ArgumentParser
Return the argparse parser of the Singularity BIDS App.

Returns **p** – Instance of `argparse.ArgumentParser`

Return type argparse.ArgumentParser

`cmp.parser.get_wrapper_parser()` → argparse.ArgumentParser
Create and return the parser object of the python wrappers of the BIDS App.

cmp.project module

Pipelines and stages modules

cmp.pipelines.common module

Definition of common parent classes for pipelines.

class `cmp.pipelines.common.Pipeline`(*project_info*)
Bases: `traits.has_traits.HasTraits`

Parent class that extends `HasTraits` and represents a processing pipeline.

It is extended by the various pipeline classes.

See also:

cmp.pipelines.anatomical.anatomical.AnatomicalPipeline, `cmp.pipelines.diffusion.diffusion.DiffusionPipeline`, *cmp.pipelines.functional.fMRI.fMRIPipeline*

anat_flow = None

clear_stages_outputs()
Clear processing stage outputs.

create_stage_flow(*stage_name*)
Create the sub-workflow of a processing stage.

Parameters **stage_name** (*str*) – Stage name

Returns **flow** – Created stage sub-workflow

Return type nipype.pipeline.engine.Workflow

fill_stages_outputs()
Update processing stage output list for visual inspection.

number_of_cores = 1

subject = 'sub-01'

cmp.pipelines.anatomical package

Submodules

cmp.pipelines.anatomical.anatomical module

Anatomical pipeline Class definition.

class `cmp.pipelines.anatomical.anatomical.`**`AnatomicalPipeline`**(*project_info*)

Bases: *`cmp.pipelines.common.Pipeline`*

Class that extends a `Pipeline` and represents the processing pipeline for structural MRI.

It is composed of the segmentation stage that performs FreeSurfer recon-all and the parcellation stage that creates the Lausanne brain parcellations.

See also:

`cmp.stages.segmentation.segmentation.SegmentationStage`, *`cmp.stages.parcellation.parcellation.ParcellationStage`*

`check_config`()

Check if custom white matter mask and custom atlas files specified in the configuration exist.

Returns **message** – String empty if all the checks pass, otherwise it contains the error message

Return type string

`check_input`(*layout*, *gui=True*)

Check if inputs of the anatomical pipeline are available.

Parameters

- **`layout`** (*`bids.BIDSLayout`*) – Instance of BIDSLayout
- **`gui`** (*`traits.Bool`*) – Boolean used to display different messages but not really meaningful anymore since the GUI components have been migrated to `cmp.bidsappmanager`

Returns **valid_inputs** – True if inputs are available

Return type traits.Bool

`check_output`()

Check if outputs of an *`AnatomicalPipeline`* are available.

Returns

- *valid_output <Bool>* – True if all outputs are found
- *error_message <string>* – Error message if an output is not found.

`create_datagrabber_node`(*base_directory*)

Create the appropriate Nipype DataGrabber node.`

Parameters **`base_directory`** (*`Directory`*) – Main CMP output directory of a subject e.g. `/output_dir/cmp/sub-XX/(ses-YY)`

Returns **datasource** – Output Nipype Node with `DataGrabber` interface

Return type Output Nipype DataGrabber Node

`create_datasinker_node`(*base_directory*)

Create the appropriate Nipype DataSink node depending on the `parcellation_scheme`

Parameters `base_directory` (*Directory*) – Main CMP output directory of a subject e.g. `/output_dir/cmp/sub-XX/(ses-YY)`

Returns **sinker** – Output Nipype Node with `DataSink` interface

Return type Output Nipype DataSink Node

`create_pipeline_flow`(*cmp_deriv_subject_directory*, *nipype_deriv_subject_directory*)

Create the pipeline workflow.

Parameters

- `cmp_deriv_subject_directory` (*Directory*) – Main CMP output directory of a subject e.g. `/output_dir/cmp/sub-XX/(ses-YY)`
- `nipype_deriv_subject_directory` (*Directory*) – Intermediate Nipype output directory of a subject e.g. `/output_dir/nipype/sub-XX/(ses-YY)`

Returns **anat_flow** – An instance of `nipype.pipeline.engine.Workflow`

Return type nipype.pipeline.engine.Workflow

`define_custom_mapping`(*custom_last_stage*)

Define the pipeline to be executed until a specific stages.

Not used yet by CMP3.

Parameters `custom_last_stage` (*string*) – Last stage to execute. Valid values are "Segmentation" and "Parcellation"

`global_conf = <cmp.pipelines.anatomical.anatomical.GlobalConfig object>`

`init_subject_derivatives_dirs`()

Return the paths to Nipype and CMP derivatives folders of a given subject / session.

Notes

`self.subject` is updated to "sub-<participant_label>_ses-<session_label>" when subject has multiple sessions.

`input_folders = ['anat']`

`now = '20221025_1350'`

`ordered_stage_list = ['Segmentation', 'Parcellation']`

`process`()

Executes the anatomical pipeline workflow and returns True if successful.

class `cmp.pipelines.anatomical.anatomical.GlobalConfig`

Bases: `traits.has_traits.HasTraits`

Global pipeline configurations.

`process_type`

Processing pipeline type

Type 'anatomical'

`subjects`

List of subjects ID (in the form `sub-XX`)

Type traits.List

`subject`
Subject to be processed (in the form `sub-XX`)

Type traits.Str

`subject_session`
Subject session to be processed (in the form `ses-YY`)

Type traits.Str

cmp.pipelines.diffusion package

Submodules

cmp.pipelines.diffusion.diffusion module

cmp.pipelines.functional package

Submodules

cmp.pipelines.functional.eeg module

EEG pipeline Class definition.

`class cmp.pipelines.functional.eeg.EEGPipeline(project_info)`
Bases: *cmp.pipelines.common.Pipeline*

Class that extends a `Pipeline` and represents the processing pipeline for EEG.

It is composed of:

- the EEG preprocessing stage that loads the input preprocessed EGG Epochs files and convert them to the MNE fif format.
- the EEG source imaging stage that takes care of all the steps necessary to extract the ROI time courses.
- the EEG connectome stage that computes different frequency- and time-frequency-domain connectivity measures from the extracted ROI time courses.

See also:

cmp.stages.eeg.preprocessing.EEGPreprocessingStage, *cmp.stages.eeg.esi.EEGSourceImagingStage*, *cmp.stages.connectome.eeg_connectome.EEGConnectomeStage*

`check_config()`

`check_input()`
Check if input of the eeg pipeline are available (Not available yet).

Returns **valid_inputs** – True if inputs are available

Return type bool

`create_datagrabber_node(name='eeg_datasource', base_directory=None, debug=False)`
Create the appropriate Nipype BIDSDataGrabber node depending on the configuration of the different EEG pipeline stages.

Parameters

- **name** (*str*) – Name of the datagrabber node

- **base_directory** (*str*) – Path to the directory that store the check_input node output
- **debug** (*bool*) – Print extra debugging messages if True

Returns **datasource** – Output Nipype Node with `BIDSDataGrabber` interface

Return type Output Nipype BIDSDataGrabber Node

create_datasinker_node(*output_directory*)

Create the appropriate Nipype DataSink node depending on EEG `task_label` and `parcellation_scheme`

Parameters **output_directory** (*Directory*) – Main CMP output directory of a subject e.g. `/output_dir/cmp/sub-XX/(ses-YY)`

Returns **sinker** – Output Nipype Node with `DataSink` interface

Return type Output Nipype DataSink Node

create_pipeline_flow(*cmp_deriv_subject_directory*, *nipype_deriv_subject_directory*)

Create the workflow of the EEG pipeline.

Parameters

- **cmp_deriv_subject_directory** (*Directory*) – Main CMP output directory of a subject e.g. `/output_dir/cmp/sub-XX/(ses-YY)`
- **nipype_deriv_subject_directory** (*Directory*) – Intermediate Nipype output directory of a subject e.g. `/output_dir/nipype/sub-XX/(ses-YY)`

Returns **eeg_flow** – An instance of `nipype.pipeline.engine.Workflow`

Return type nipype.pipeline.engine.Workflow

get_nipype_eeg_pipeline_subject_dir()

Return the path to Nipype eeg_pipeline folder of a given subject / session.

global_conf = <cmp.pipelines.functional.eeg.GlobalConfig object>

init_subject_derivatives_dirs()

Return the paths to Nipype and CMP derivatives folders of a given subject / session.

Notes

`self.subject` is updated to "sub-<participant_label>_ses-<session_label>" when subject has multiple sessions.

input_folders = ['anat', 'eeg']

now = '20221025_1350'

ordered_stage_list = ['EEGPreparer', 'EEGLoader', 'InverseSolution']

process()

Executes the anatomical pipeline workflow and returns True if successful.

class cmp.pipelines.functional.eeg.**GlobalConfig**

Bases: `traits.has_traits.HasTraits`

Global EEG pipeline configurations.

process_type

Processing pipeline type

Type 'EEG'

subjects
List of subjects ID (in the form `sub-XX`)

Type traits.List

subject
Subject to be processed (in the form `sub-XX`)

Type traits.Str

subject_session
Subject session to be processed (in the form `ses-YY`)

Type traits.Str

cmp.pipelines.functional.fMRI module

Functional pipeline Class definition.

class `cmp.pipelines.functional.fMRI.`**GlobalConfig**
Bases: `traits.has_traits.HasTraits`

Global pipeline configurations.

process_type
Processing pipeline type

Type 'fMRI'

imaging_model
Imaging model used by `RegistrationStage`

Type 'fMRI'

class `cmp.pipelines.functional.fMRI.`**fMRIPipeline**(*project_info*)
Bases: *cmp.pipelines.common.Pipeline*

Class that extends a `Pipeline` and represents the processing pipeline for structural MRI.

It is composed of:

- the preprocessing stage that can perform slice timing correction, deskiping and motion correction
- the registration stage that co-registered the anatomical T1w scan to the mean BOLD image and projects the parcellations to the native fMRI space
- the extra-preprocessing stage (FunctionalMRIStage) that can perform nuisance regression and band-pass filtering
- the connectome stage that extracts the time-series of each parcellation ROI and computes the Pearson's correlation coefficient between ROI time-series to create the functional connectome.

See also:

cmp.stages.preprocessing.fmri_preprocessing.PreprocessingStage, *cmp.stages.registration.registration.RegistrationStage*, *cmp.stages.functional.functionalMRI.FunctionalMRIStage*, *cmp.stages.connectome.fmri_connectome.ConnectomeStage*

check_config()
Check if the fMRI pipeline parameters is properly configured.

Returns **message** – String that is empty if success, otherwise it contains the error message

Return type string

check_input(*layout*, *gui=True*)

Check if input of the diffusion pipeline are available.

Parameters

- **layout** (*bids.BIDSLayout*) – Instance of BIDSLayout
- **gui** (*traits.Bool*) – Boolean used to display different messages but not really meaningful anymore since the GUI components have been migrated to `cmp.bidsappmanager`

Returns **valid_inputs** – True if inputs are available

Return type traits.Bool

create_datagrabber_node(*base_directory*, *bids_atlas_label*)

Create the appropriate Nipype DataGrabber node depending on the `parcellation_scheme`

Parameters

- **base_directory** (*Directory*) – Main CMP output directory of a subject e.g. `/output_dir/cmp/sub-XX/(ses-YY)`
- **bids_atlas_label** (*string*) – Parcellation atlas label

Returns **datasource** – Output Nipype Node with `DataGrabber` interface

Return type Output Nipype DataGrabber Node

create_datasinker_node(*base_directory*, *bids_atlas_label*)

Create the appropriate Nipype DataSink node depending on the `parcellation_scheme`

Parameters

- **base_directory** (*Directory*) – Main CMP output directory of a subject e.g. `/output_dir/cmp/sub-XX/(ses-YY)`
- **bids_atlas_label** (*string*) – Parcellation atlas label

Returns **sinker** – Output Nipype Node with `DataSink` interface

Return type Output Nipype DataSink Node

create_pipeline_flow(*cmp_deriv_subject_directory*, *nipype_deriv_subject_directory*)

Create the pipeline workflow.

Parameters

- **cmp_deriv_subject_directory** (*Directory*) – Main CMP output directory of a subject e.g. `/output_dir/cmp/sub-XX/(ses-YY)`
- **nipype_deriv_subject_directory** (*Directory*) – Intermediate Nipype output directory of a subject e.g. `/output_dir/nipype/sub-XX/(ses-YY)`

Returns **fMRI_flow** – An instance of `nipype.pipeline.engine.Workflow`

Return type nipype.pipeline.engine.Workflow

define_custom_mapping(*custom_last_stage*)

Define the pipeline to be executed until a specific stages.

Not used yet by CMP3.

Parameters **custom_last_stage** (*string*) – Last stage to execute. Valid values are: "Preprocessing", "Registration", "FunctionalMRI" and "Connectome".

global_conf = <cmp.pipelines.functional.fMRI.GlobalConfig object>

init_subject_derivatives_dirs()
Return the paths to Nipype and CMP derivatives folders of a given subject / session.

Notes

`self.subject` is updated to "sub-<participant_label>_ses-<session_label>" when subject has multiple sessions.

input_folders = ['anat', 'func']

now = '20221025_1350'

ordered_stage_list = ['Preprocessing', 'Registration', 'FunctionalMRI', 'Connectome']

process()
Executes the fMRI pipeline workflow and returns True if successful.

update_nuisance_requirements()
Update nuisance requirements.

Configure the registration to apply the estimated transformation to multiple segmentation masks depending on the Nuisance correction steps performed.

update_registration()
Configure the list of registration tools.

update_scrubbing()
Update to precompute or inputs for scrubbing during the FunctionalMRI stage.

cmp.stages package

Subpackages

cmp.stages.connectome package

Submodules

cmp.stages.connectome.connectome module

Definition of config and stage classes for building structural connectivity matrices.

class `cmp.stages.connectome.connectome.`**ConnectomeConfig**
Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a *ConnectomeStage* instance.

compute_curvature
Compute fiber curvature (Default: False)

Type traits.Bool

output_types
Output connectome format

Type ['gpickle', 'mat', 'graphml']

connectivity_metrics
Set of connectome maps to compute

Type ['Fiber number', 'Fiber length', 'Fiber density', 'Fiber proportion', 'Normalized fiber density', 'ADC', 'gFA']

log_visualization
Log visualization that might be obsolete as this has been detached after creation of the bidsappmanager (Default: True)

Type traits.Bool

circular_layout
Visualization of the connectivity matrix using a circular layout that might be obsolete as this has been detached after creation of the bidsappmanager (Default: False)

Type traits.Bool

subject
BIDS subject ID (in the form `sub-XX`)

Type traits.Str

See also:

cmp.stages.connectome.connectome.ConnectomeStage

class cmp.stages.connectome.connectome.**ConnectomeStage**(*bids_dir*, *output_dir*)
Bases: *cmp.stages.common.Stage*

Class that represents the connectome building stage of a `DiffusionPipeline`.

create_workflow()
Create the workflow of the diffusion *ConnectomeStage*

See also:

`cmp.pipelines.diffusion.diffusion.DiffusionPipeline`, *cmp.stages.connectome.connectome.ConnectomeConfig*

create_workflow(*flow*, *inputnode*, *outputnode*)
Create the stage workflow.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the Diffusion pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

define_inspect_outputs()
Update the `inspect_outputs` class attribute.

It contains a dictionary of stage outputs with corresponding commands for visual inspection.

cmp.stages.connectome.eeg_connectome module

Definition of config and stage classes for building functional connectivity matrices from preprocessed EEG.

class `cmp.stages.connectome.eeg_connectome.`**`EEGConnectomeConfig`**

Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a *EEGConnectomeStage* instance.

`task_label`

Task label (e.g. `_task-<label>_`)

Type Str

`parcellation_scheme`

Parcellation used to create the ROI source time-series

Type Enum([“NativeFreesurfer”, “Lausanne2018”])

`lausanne2018_parcellation_res`

Resolution of the parcellation if Lausanne2018 parcellation scheme is used

Type Enum([“scale1”, “scale2”, “scale3”, “scale4”, “scale5”])

`connectivity_metrics`

Set of frequency- and time-frequency-domain connectivity metrics to compute

Type [‘coh’, ‘cohy’, ‘imcoh’, ‘plv’, ‘ciplv’, ‘ppc’, ‘pli’, ‘wpli’, ‘wpli2_debiased’]

`output_types`

Output connectome file format

Type [‘tsv’, ‘gpickle’, ‘mat’, ‘graphml’]

See also:

cmp.stages.connectome.eeg_connectome.EEGConnectomeStage

class `cmp.stages.connectome.eeg_connectome.`**`EEGConnectomeStage`**(*bids_dir*, *output_dir*, *subject*, *session=''*)

Bases: *cmp.stages.common.Stage*

Class that represents the connectome building stage of a *EEGPipeline*.

See also:

cmp.pipelines.functional.eeg.EEGPipeline, *cmp.stages.connectome.eeg_connectome.EEGConnectomeConfig*

`create_workflow`(*flow*, *inputnode*, *outputnode*)

Create the stage workflow.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the EEG pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

`define_inspect_outputs`(*log_visualization=True*, *circular_layout=False*)

Update the `inspect_outputs` class attribute.

It contains a dictionary of stage outputs with corresponding commands for visual inspection.

cmp.stages.connectome.fmri_connectome module

Definition of config and stage classes for building functional connectivity matrices.

class cmp.stages.connectome.fmri_connectome.**ConnectomeConfig**

Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a *ConnectomeStage* instance.

apply_scrubbing

Apply scrubbing before mapping the functional connectome if True (Default: False)

Type traits.Bool

FD_thr

Framewise displacement threshold (Default: 0.2)

Type traits.Float

DVARS_thr

DVARS (RMS of variance over voxels) threshold (Default: 4.0)

Type traits.Float

output_types

Output connectome format

Type ['gpickle', 'mat', 'cff', 'graphml']

log_visualization

Log visualization that might be obsolete as this has been detached after creation of the bidsappmanager (Default: True)

Type traits.Bool

circular_layout

Visualization of the connectivity matrix using a circular layout that might be obsolete as this has been detached after creation of the bidsappmanager (Default: False)

Type traits.Bool

subject

BIDS subject ID (in the form `sub-XX`)

Type traits.Str

See also:

cmp.stages.connectome.fmri_connectome.ConnectomeStage

class cmp.stages.connectome.fmri_connectome.**ConnectomeStage**(*bids_dir*, *output_dir*)

Bases: *cmp.stages.common.Stage*

Class that represents the connectome building stage of a *fMRIPipeline*.

create_workflow()

Create the workflow of the fMRI *ConnectomeStage*

See also:

cmp.pipelines.functional.fMRI.fMRIPipeline, *cmp.stages.connectome.fmri_connectome.ConnectomeConfig*

create_workflow(*flow*, *inputnode*, *outputnode*)

Create the stage worflow.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the fMRI pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

define_inspect_outputs()

Update the `inspect_outputs` class attribute.

It contains a dictionary of stage outputs with corresponding commands for visual inspection.

cmp.stages.diffusion package

Submodules

cmp.stages.diffusion.diffusion module

cmp.stages.diffusion.reconstruction module

cmp.stages.diffusion.tracking module

cmp.stages.eeg package

Submodules

cmp.stages.eeg.esi module

Definition of config and stage classes for computing brain parcellation.

class cmp.stages.eeg.esi.**EEGSourceImagingConfig**

Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a *EEGSourceImagingStage* instance.

task_label

Task label (e.g. `_task-<label>_`)

Type Str

esi_tool

Select the tool used for EEG source imaging (inverse solution)

Type Enum

mne_apply_electrode_transform

If `True`, apply the transform specified below to electrode positions

Type Bool

mne_electrode_transform_file
Instance of CustomEEGCartoolMNETransformBIDSFile that describes the input BIDS-formatted MNE transform file in fif format

Type *CustomEEGMNETransformBIDSFile*

cartool_spi_file
Instance of *CustomEEGCartoolSpiBIDSFile* that describes the input BIDS-formatted EEG Solution Points Irregularly spaced file created by Cartool

Type *CustomEEGCartoolSpiBIDSFile*

cartool_invsol_file
Instance of *CustomEEGCartoolInvSolBIDSFile* that describes the input BIDS-formatted EEG Inverse Solution file created by Cartool

Type *CustomEEGCartoolInvSolBIDSFile*

cartool_esi_method
Cartool Source Imaging method

Type Enum([‘LAURA’, ‘LORETA’])

parcellation_scheme
Parcellation used to create the ROI source time-series

Type Enum([“NativeFreesurfer”, “Lausanne2018”])

lausanne2018_parcellation_res
Resolution of the parcellation if Lausanne2018 parcellation scheme is used

Type Enum([“scale1”, “scale2”, “scale3”, “scale4”, “scale5”])

cartool_esi_lamb
Regularization weight of inverse solutions computed with Cartool (Default: 6)

Type Float

cartool_svd_toi_begin
Start TOI for SVD projection (Default: 0.0)

Type Float

cartool_svd_toi_end
End TOI for SVD projection (Default: 0.25)

Type Float

mne_esi_method
MNE Source Imaging method

Type Enum([“sLORETA”, “eLORETA”, “MNE”, “dSPM”])

mne_esi_method_snr
SNR value such as the regularization weight lambda2 of MNE ESI method’ is set to `1.0 / mne_esi_method_snr ** 2` (Default: 3.0)

Type Float

See also:

cmp.stages.eeg.esi.EEGSourceImagingStage

class cmp.stages.eeg.esi.**EEGSourceImagingStage**(*subject*, *session*, *bids_dir*, *output_dir*)
Bases: *cmp.stages.common.Stage*

Class that represents the reconstruction of the inverse solutions stage of a *EEGPipeline*.

If MNE is selected for ESI reconstruction, this stage consists of five processing interfaces:

- `CreateBem`: Create the Boundary Element Model that consists of surfaces obtained with Freesurfer.
- `CreateSrc`: Create a bilateral hemisphere surface-based source space file with subsampling.
- `CreateFwd`: Create the forward solution (leadfield) from the BEM and the source space.
- `CreateCov`: Create the noise covariance matrix from the data.
- `MNEInverseSolutionROI`: Create and apply the actual inverse operator to generate the ROI time courses.

If you decide to use ESI reconstruction outputs precomputed with Cartool, then this stage consists of two processing interfaces:

- `CreateSpiRoisMapping`: Create Cartool-reconstructed sources / parcellation ROI mapping file.
- `CartoolInverseSolutionROIExtraction`: Use Pycartool to load inverse solutions estimated by Cartool and generate the ROI time courses.

See also:

cmp.pipelines.functional.eeg.EEGPipeline, *cmp.stages.eeg.esi.EEGSourceImagingConfig*

create_cartool_workflow(*flow*, *inputnode*, *outputnode*)

Create the stage workflow using Cartool-precomputed inverse solutions.

This method is called by `create_workflow()` main function if Cartool is selected for ESI.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the EEG pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

create_mne_workflow(*flow*, *inputnode*, *outputnode*)

Create the stage workflow using MNE.

This method is called by `create_workflow()` main function if MNE is selected for ESI.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the EEG pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

create_workflow(*flow*, *inputnode*, *outputnode*)

Main method to create the stage workflow.

Based on the tool used for ESI, this method calls either the `create_cartool_workflow()` or the :func:`~cmp.stages.eeg.esi.create_mne_workflow method.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the EEG pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

define_inspect_outputs()

Update the `inspect_outputs` class attribute.

It contains a dictionary of stage outputs with corresponding commands for visual inspection.

cmp.stages.eeg.preprocessing module

Definition of config and stage classes for computing brain parcellation.

class cmp.stages.eeg.preprocessing.**EEGPreprocessingConfig**

Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a `EEGPreprocessingStage` instance.

task_label

Task label (e.g. `_task-<label>_`)

Type Str

eeg_ts_file

Instance of *CustomEEGPreprocBIDSFile* that describes the input BIDS-formatted preprocessed EEG file

Type *CustomEEGPreprocBIDSFile*

events_file

Instance of *CustomEEGEventsBIDSFile* that describes the input BIDS-formatted EEG events file

Type *CustomEEGEventsBIDSFile*

electrodes_file_fmt

Select the type of tabular file describing electrode positions

Type Enum([“BIDS”, “Cartool”])

bids_electrodes_file

Instance of *CustomEEGElectrodesBIDSFile* that describes the input BIDS-compliant EEG electrode file

Type *CustomEEGElectrodesBIDSFile*

cartool_electrodes_file

Instance of *CustomEEGCartoolElectrodesBIDSFile* that describes the input BIDS-formatted EEG electrode file created by Cartool

Type *CustomEEGCartoolElectrodesBIDSFile*

t_min

Start time of the epochs in seconds, relative to the time-locked event (Default: -0.2)

Type Float

t_max

End time of the epochs in seconds, relative to the time-locked event (Default: 0.5)

Type Float

See also:

cmp.stages.eeg.preparer.EEGPreprocessingStage

class cmp.stages.eeg.preprocessing.**EEGPreprocessingStage**(*subject*, *session*, *bids_dir*, *output_dir*)

Bases: *cmp.stages.common.Stage*

Class that represents the preprocessing stage of a *EEGPipeline*.

This stage consists of converting EEGLab `set` EEG files to MNE Epochs in `fif` format, the format used in the rest of the pipeline by calling, if necessary the following interface:

- `EEGLAB2fif`: Reads eeglab data and converts them to MNE format (`fif` file extension).

See also:

cmp.pipelines.functional.eeg.EEGPipeline, cmp.stages.eeg.preparer.EEGPreprocessingConfig

create_workflow(*flow*, *inputnode*, *outputnode*)

Create the stage workflow.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the EEG pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

define_inspect_outputs()

Update the `inspect_outputs` class attribute.

It contains a dictionary of stage outputs with corresponding commands for visual inspection.

cmp.stages.functional package

Submodules

cmp.stages.functional.functionalMRI module

Definition of config and stage classes for the extra functional preprocessing stage.

class cmp.stages.functional.functionalMRI.**FunctionalMRIConfig**

Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a `FunctionalMRIStage` object.

global_nuisance

Perform global nuisance regression (Default: False)

Type traits.Bool

csf

Perform CSF nuisance regression (Default: True)

Type traits.Bool

wm

Perform White-Matter nuisance regression (Default: True)

Type traits.Bool

motion
Perform motion nuisance regression (Default: True)

Type traits.Bool

detrending = Bool
Perform detrending (Default: True)

detrending_mode = Enum("linear", "quadratic")
Detrending mode (Default: "Linear")

bandpass_filtering = Bool
Perform bandpass filtering (Default: True)

lowpass_filter = Float
Lowpass filter frequency (Default: 0.01)

highpass_filter = Float
Highpass filter frequency (Default: 0.1)

scrubbing = Bool
Perform scrubbing (Default: True)

See also:

cmp.stages.functional.functionalMRI.FunctionalMRIStage

class cmp.stages.functional.functionalMRI.**FunctionalMRIStage**(*bids_dir*, *output_dir*)
Bases: *cmp.stages.common.Stage*

Class that represents the post-registration preprocessing stage of the fMRIPipeline.

create_workflow()
Create the workflow of the *FunctionalMRIStage*

See also:

cmp.pipelines.functional.fMRI.fMRIPipeline, *cmp.stages.functional.functionalMRI.FunctionalMRIConfig*

create_workflow(*flow*, *inputnode*, *outputnode*)
Create the stage worflow.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the fMRI pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

define_inspect_outputs()
Update the inspect_outputs class attribute.

It contains a dictionary of stage outputs with corresponding commands for visual inspection.

cmp.stages.parcellation package

Submodules

cmp.stages.parcellation.parcellation module

Definition of config and stage classes for computing brain parcellation.

class `cmp.stages.parcellation.parcellation.ParcellationConfig`

Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a *ParcellationStage* object.

pipeline_mode

Distinguish if a parcellation is run in a "Diffusion" or in a fMRI pipeline

Type traits.Enum(["Diffusion", "fMRI"])

parcellation_scheme

Parcellation scheme used (Default: 'Lausanne2018')

Type traits.Str

parcellation_scheme_editor

Choice of parcellation schemes

Type traits.List(['NativeFreesurfer', 'Lausanne2018', 'Custom'])

include_thalamic_nuclei_parcellation

Perform and include thalamic nuclei segmentation in 'Lausanne2018' parcellation (Default: True)

Type traits.Bool

ants_precision_type

Specify ANTs used by thalamic nuclei segmentation to adopt single / double precision float representation to reduce memory usage. (Default: 'double')

Type traits.Enum(['double', 'float'])

segment_hippocampal_subfields

Perform and include FreeSurfer hippocampal subfields segmentation in 'Lausanne2018' parcellation (Default: True)

Type traits.Bool

segment_brainstem

Perform and include FreeSurfer brainstem segmentation in 'Lausanne2018' parcellation (Default: True)

Type traits.Bool

atlas_info

Dictionary storing information of atlases in the form >>> atlas_info = { >>> "atlas_name": { >>> 'number_of_regions': 83, >>> 'node_information_graphml': "/path/to/file.graphml" >>> } >>> } # doctest: +SKIP

Type traits.Dict

custom_parcellation

Instance of *CustomParcellationBIDSFile* that describes the custom BIDS-formatted brain parcellation file

Type traits.Instance(*CustomParcellationBIDSFile*)

See also:

cmp.stages.parcellation.parcellation.ParcellationStage

class cmp.stages.parcellation.parcellation.**ParcellationStage**(*pipeline_mode*, *subject*, *session*, *bids_dir*, *output_dir*)

Bases: *cmp.stages.common.Stage*

Class that represents the parcellation stage of a AnatomicalPipeline.

create_workflow()

Create the workflow of the ParcellationStage

See also:

cmp.pipelines.anatomical.anatomical.AnatomicalPipeline, *cmp.stages.parcellation.parcellation.ParcellationConfig*

create_workflow(*flow*, *inputnode*, *outputnode*)

Create the stage workflow.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the anatomical pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the parcellation stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the parcellation stage

create_workflow_custom(*flow*, *outputnode*)

Create the stage workflow when custom inputs are specified.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the anatomical pipeline
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the parcellation stage

define_inspect_outputs()

Update the inspect_outputs class attribute.

It contains a dictionary of stage outputs with corresponding commands for visual inspection.

cmp.stages.preprocessing package

Submodules

cmp.stages.preprocessing.fmri_preprocessing module

Definition of config and stage classes for pre-registration fMRI preprocessing.

class cmp.stages.preprocessing.fmri_preprocessing.**PreprocessingConfig**

Bases: traits.has_traits.HasTraits

Class used to store configuration parameters of a *PreprocessingStage* object.

discard_n_volumes
(Default: '5')

Type traits.Int

despiking
(Default: True)

Type traits.Bool

slice_timing
Slice acquisition order for slice timing correction that can be: "bottom-top interleaved", "bottom-top interleaved", "top-bottom interleaved", "bottom-top", and "top-bottom" (Default: "none")

Type traits.Enum

repetition_time
Repetition time (Default: 1.92)

Type traits.Float

motion_correction
Perform motion correction (Default: True)

Type traits.Bool

See also:

cmp.stages.preprocessing.fmri_preprocessing.PreprocessingStage

class cmp.stages.preprocessing.fmri_preprocessing.**PreprocessingStage**(*bids_dir*, *output_dir*)
Bases: *cmp.stages.common.Stage*

Class that represents the pre-registration preprocessing stage of a *fMRIPipeline* instance.

create_workflow()
Create the workflow of the *PreprocessingStage*

See also:

cmp.pipelines.functional.fMRI.fMRIPipeline, *cmp.stages.preprocessing.fmri_preprocessing.PreprocessingConfig*

create_workflow(*flow*, *inputnode*, *outputnode*)
Create the stage worflow.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the fMRI pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

define_inspect_outputs()
Update the `inspect_outputs` class attribute.

It contains a dictionary of stage outputs with corresponding commands for visual inspection.

cmp.stages.preprocessing.preprocessing module

Definition of config and stage classes for diffusion MRI preprocessing.

class `cmp.stages.preprocessing.preprocessing.`**`PreprocessingConfig`**

Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a *PreprocessingStage* instance.

`total_readout`

Acquisition total readout time used by FSL Eddy (Default: 0.0)

Type traits.Float

`description`

Description (Default: 'description')

Type traits.Str

`denoising`

Perform diffusion MRI denoising (Default: False)

Type traits.Bool

`denoising_algo`

Type of denoising algorithm (Default: 'MRtrix (MP-PCA)')

Type traits.Enum(['MRtrix (MP-PCA)', 'Dipy (NLM)'])

`dipy_noise_model`

Type of noise model when Dipy denoising is performed that can be: 'Rician' or 'Gaussian' (Default: 'Rician')

Type traits.Enum

`bias_field_correction`

Perform diffusion MRI bias field correction (Default: False)

Type traits.Bool

`bias_field_algo`

Type of bias field correction algorithm that can be: 'ANTS N4' or 'FSL FAST' (Default: 'ANTS N4')

Type traits.Enum, ['ANTS N4', 'FSL FAST'])

`eddy_current_and_motion_correction`

Perform eddy current and motion correction (Default: True)

Type traits.Bool

`eddy_correction_algo`

Algorithm used for eddy current correction that can be: 'FSL eddy_correct' or 'FSL eddy' (Default: 'FSL eddy_correct')

Type traits.Enum

`eddy_correct_motion_correction`

Perform eddy current and motion correction MIGHT BE OBSOLETE (Default: True)

Type traits.Bool

`partial_volume_estimation`

Estimate partial volume maps from brain tissues segmentation (Default: True)

Type traits.Bool

fast_use_priors
Use priors when FAST is used for partial volume estimation (Default: True)

Type traits.Bool

resampling
Tuple describing the target resolution (Default: (1, 1, 1))

Type traits.Tuple

interpolation
Type of interpolation used when resampling that can be: 'interpolate', 'weighted', 'nearest', 'sinc', or 'cubic' (Default: 'interpolate')

Type traits.Enum

tracking_tool
Tool used for tractography

Type Enum(['Dipy', 'MRtrix'])

act_tracking
True if Anatomically-Constrained or Particle Filtering Tractography is enabled (Default: False)

Type Bool

gmwmi_seeding
True if tractography seeding is performed from the gray-matter / white-matter interface (Default: False)

Type Bool

See also:

cmp.stages.preprocessing.preprocessing.PreprocessingStage

class `cmp.stages.preprocessing.preprocessing.`**PreprocessingStage**(*bids_dir*, *output_dir*)
Bases: *cmp.stages.common.Stage*

Class that represents the pre-registration preprocessing stage of a `DiffusionPipeline` instance.

create_workflow()
Create the workflow of the *PreprocessingStage*

See also:

`cmp.pipelines.diffusion.diffusion.DiffusionPipeline`, *cmp.stages.preprocessing.preprocessing.PreprocessingConfig*

create_workflow(*flow*, *inputnode*, *outputnode*)
Create the stage worflow.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the Diffusion pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

define_inspect_outputs()
Update the `inspect_outputs` class attribute.

It contains a dictionary of stage outputs with corresponding commands for visual inspection.

cmp.stages.registration package

Submodules

cmp.stages.registration.registration module

Definition of config and stage classes for MRI co-registration.

class `cmp.stages.registration.registration.`**`RegistrationConfig`**

Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a *RegistrationStage* instance.

`pipeline`

Pipeline type (Default: "Diffusion")

Type traits.Enum(["Diffusion", "fMRI"])

`registration_mode_trait`

Choices of registration tools updated depending on the `pipeline` type. (Default: ['FSL', 'ANTs'] if "Diffusion", ['FSL', 'BBregister (FS)'] if "fMRI")

Type traits.List(['FSL', 'ANTs', 'BBregister (FS)'])

`registration_mode`

Registration tool used from the *registration_mode_trait* list (Default: 'ANTs')

Type traits.Str

`diffusion_imaging_model`

Diffusion imaging model ('DTI' for instance)

Type traits.Str

`use_float_precision`

Use 'single' instead of 'double' float representation to reduce memory usage of ANTs (Default: False)

Type traits.Bool

`ants_interpolation`

Interpolation type used by ANTs that can be: 'Linear', 'NearestNeighbor', 'CosineWindowedSinc', 'WelchWindowedSinc', 'HammingWindowedSinc', 'LanczosWindowedSinc', 'BSpline', 'MultiLabel', or 'Gaussian' (Default: 'Linear')

Type traits.Enum

`ants_bspline_interpolation_parameters`

ANTs BSpline interpolation parameters (Default: traits.Tuple(Int(3)))

Type traits.Tuple

`ants_gauss_interpolation_parameters`

ANTs Gaussian interpolation parameters (Default: traits.Tuple(Float(5), Float(5)))

Type traits.Tuple

`ants_multilab_interpolation_parameters`

ANTs Multi-label interpolation parameters (Default: traits.Tuple(Float(5), Float(5)))

Type traits.Tuple

`ants_lower_quantile`

ANTs lower quantile (Default: 0.005)

Type traits.Float

`ants_upper_quantile`
ANTs upper quantile (Default: 0.995)

Type traits.Float

`ants_convergence_thresh`
ANTs convergence threshold (Default: 1e-06)

Type traits.Float

`ants_convergence_winsize`
ANTs convergence window size (Default: 10)

Type traits.Int

`ants_linear_gradient_step`
ANTS linear gradient step size (Default: 0.1)

Type traits.Float

`ants_linear_cost`
Metric used by ANTs linear registration phase that can be 'CC', 'MeanSquares', 'Demons', 'GC', 'MI', or 'Mattes' (Default: 'MI')

Type traits.Enum

`ants_linear_sampling_strategy`
ANTS sampling strategy for the linear registration phase that can be 'None', 'Regular', or 'Random' (Default: 'Regular')

Type traits.Enum

`ants_linear_sampling_perc`
Percentage used if random sampling strategy is employed in the linear registration phase (Default: 0.25)

Type traits.Float

`ants_perform_syn`
(Default: True)

Type traits.Bool

`ants_nonlinear_gradient_step`
(Default: 0.1)

Type traits.Float

`ants_nonlinear_cost`
Metric used by ANTs nonlinear (SyN) registration phase that can be 'CC', 'MeanSquares', 'Demons', 'GC', 'MI', or 'Mattes' (Default: 'CC')

Type traits.Enum

`ants_nonlinear_update_field_variance`
Weight to update field variance in ANTs nonlinear (SyN) registration phase (Default: 3.0)

Type traits.Float

`ants_nonlinear_total_field_variance`
Weight to give to total field variance in ANTs nonlinear (SyN) registration phase (Default: 0.0)

Type traits.Float

flirt_args
FLIRT extra arguments that will be append to the FSL FLIRT command (Default: None)

Type traits.Str

uses_qform
FSL FLIRT uses qform (Default: True)

Type traits.Bool

dof
Specify number of degree-of-freedom to FSL FLIRT (Default: 6)

Type traits.Int

fsl_cost
Metric used by FSL registration that can be 'mutualinfo', 'corratio', 'normcorr', 'normmi', 'leastsq', or 'labeldiff' (Default: 'normmi')

Type traits.Enum

no_search
Enable FSL FLIRT "no search" option (Default: True)

Type traits.Bool

init
Initialization type of FSL registration: 'spm', 'fsl', or 'header' (Default: 'smp')

Type traits.Enum('header', ['spm', 'fsl', 'header'])

contrast_type
Contrast type specified to BBRegister: 't1', 't2', or 'dti' (Default: 'dti')

Type traits.Enum('dti', ['t1', 't2', 'dti'])

apply_to_eroded_wm
Apply estimated transform to eroded white-matter mask (Default: True)

Type traits.Bool

apply_to_eroded_csf
Apply estimated transform to eroded cortico spinal fluid mask (Default: True)

Type traits.Bool

apply_to_eroded_brain
Apply estimated transform to eroded brain mask (Default: False)

Type traits.Bool

tracking_tool
Tool used for tractography

Type Enum(['Dipy', 'MRtrix'])

act_tracking
True if Anatomically-Constrained or Particle Filtering Tractography is enabled (Default: False)

Type traits.Bool

gmwmi_seeding
True if tractography seeding is performed from the gray-matter / white-matter interface (Default: False)

Type traits.Bool

See also:

cmp.stages.registration.registration.RegistrationStage

class cmp.stages.registration.registration.**RegistrationStage**(*pipeline_mode*, *fs_subjects_dir=None*, *fs_subject_id=None*, *bids_dir=''*, *output_dir=''*)

Bases: *cmp.stages.common.Stage*

Class that represents the registration stage of both DiffusionPipeline and fMRIPipeline.

fs_subjects_dir

Freesurfer subjects directory (needed by BBRegister)

Type traits.Directory

fs_subject_id

Freesurfer subject (being processed) directory (needed by BBRegister)

Type traits.Str

create_workflow()

Create the workflow of the *RegistrationStage*

See also:

cmp.pipelines.diffusion.diffusion.DiffusionPipeline, *cmp.pipelines.functional.fMRI.fMRIPipeline*, *cmp.stages.registration.registration.RegistrationConfig*

create_ants_workflow(*flow*, *inputnode*, *outputnode*)

Create the registration workflow using ANTs.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the Diffusion pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

create_bbregister_workflow(*flow*, *inputnode*, *outputnode*)

Create the workflow of the registration stage using FreeSurfer BBRegister.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the fMRI pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

create_flirt_workflow(*flow*, *inputnode*, *outputnode*)

Create the workflow of the registration stage using FSL FLIRT.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of either the Diffusion pipeline or the fMRI pipeline

- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

create_workflow(*flow*, *inputnode*, *outputnode*)
Create the stage workflow.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of either the Diffusion pipeline or the fMRI pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the stage

define_inspect_outputs()
Update the `inspect_outputs` class attribute.

It contains a dictionary of stage outputs with corresponding commands for visual inspection.

cmp.stages.segmentation package

Submodules

cmp.stages.segmentation.segmentation module

Definition of config and stage classes for segmentation.

class cmp.stages.segmentation.segmentation.**SegmentationConfig**
Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a *SegmentationStage* object.

seg_tool
Choice of segmentation tool that can be “Freesurfer”

Type traits.Enum([“Freesurfer”, “Custom segmentation”])

make_isotropic
Resample to isotropic resolution (Default: False)

Type traits.Bool

isotropic_vox_size
Isotropic resolution to be resampled (Default: 1.2, desc=”)

Type traits.Float

isotropic_interpolation
Interpolation type used for resampling that can be: ‘cubic’, ‘weighted’, ‘nearest’, ‘sinc’, or ‘interpolate’, (Default: ‘cubic’)

Type traits.Enum

brain_mask_extraction_tool
Choice of brain extraction tool: “Freesurfer”, “BET”, or “ANTs” (Default: Freesurfer)

Type traits.Enum

`ants_templatefile`
Anatomical template used by ANTS brain extraction

Type traits.File

`ants_probmaskfile`
Brain probability mask used by ANTS brain extraction

Type traits.File

`ants_regmaskfile`
Mask (defined in the template space) used during registration in ANTs brain extraction. To limit the metric computation to a specific region.

Type traits.File

`use_fsl_brain_mask`
Use FSL BET for brain extraction (Default: False)

Type traits.Bool

`use_existing_freesurfer_data`
(Default: False)

Type traits.Bool

`freesurfer_subjects_dir`
Freesurfer subjects directory path usually `/output_dir/freesurfer`

Type traits.Str

`freesurfer_subject_id`
Freesurfer subject (being processed) ID in the form `sub-XX(_ses-YY)`

Type traits.Str

`freesurfer_args`
Extra Freesurfer `recon-all` arguments

Type traits.Str

`custom_brainmask`
Instance of `CustomBrainMaskBIDSFile` that describes the custom BIDS formatted brain mask

Type traits.Instance(*CustomBrainMaskBIDSFile*)

`custom_wm_mask`
Instance of `CustomWMMaskBIDSFile` that describes the custom BIDS formatted white-matter mask

Type traits.Instance(*CustomWMMaskBIDSFile*)

`custom_gm_mask`
Instance of `CustomGMMaskBIDSFile` that describes the custom BIDS formatted gray-matter mask

Type traits.Instance(*CustomGMMaskBIDSFile*)

`custom_csf_mask`
Instance of `CustomCSFMaskBIDSFile` that describes the custom BIDS formatted CSF mask

Type traits.Instance(*CustomCSFMaskBIDSFile*)

`custom_aparcaseg`
Instance of `CustomAparcAsegBIDSFile` that describes the custom BIDS formatted Freesurfer aparc-aseg file

Type traits.Instance(*CustomAparcAsegBIDSFile*)

number_of_threads
Number of threads leveraged by OpenMP and used in the stage by Freesurfer and ANTs (Default: 1)

Type traits.Int

See also:

cmp.stages.segmentation.segmentation.SegmentationStage, *cmtklib.bids.io.CustomBrainMaskBIDSFile*, *cmtklib.bids.io.CustomWMMaskBIDSFile*, *cmtklib.bids.io.CustomGMMaskBIDSFile*, *cmtklib.bids.io.CustomCSFMaskBIDSFile*

class cmp.stages.segmentation.segmentation.**SegmentationStage**(*subject*, *session*, *bids_dir*, *output_dir*)

Bases: *cmp.stages.common.Stage*

Class that represents the segmentation stage of a *AnatomicalPipeline*.

create_workflow()
Create the workflow of the *SegmentationStage*

See also:

cmp.pipelines.anatomical.anatomical.AnatomicalPipeline, *cmp.stages.segmentation.segmentation.SegmentationConfig*

create_workflow(*flow*, *inputnode*, *outputnode*)
Create the stage workflow.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the anatomical pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the segmentation stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the segmentation stage

create_workflow_custom(*flow*, *inputnode*, *outputnode*)
Create the stage workflow when custom inputs are specified.

Parameters

- **flow** (*nipype.pipeline.engine.Workflow*) – The nipype.pipeline.engine.Workflow instance of the anatomical pipeline
- **inputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the inputs of the segmentation stage
- **outputnode** (*nipype.interfaces.utility.IdentityInterface*) – Identity interface describing the outputs of the segmentation stage

define_inspect_outputs(*debug=False*)
Update the `inspect_outputs` class attribute.

It contains a dictionary of stage outputs with corresponding commands for visual inspection.

Parameters **debug** (*bool*) – If `True`, show printed output

Submodules

cmp.stages.common module

Definition of common parent classes for stages.

class `cmp.stages.common.Stage`

Bases: `traits.has_traits.HasTraits`

Parent class that extends `HasTraits` and represents a processing pipeline stage.

It is extended by the various pipeline stage subclasses.

bids_subject_label
BIDS subject (participant) label

Type traits.Str

bids_session_label
BIDS session label

Type traits.Str

bids_dir
BIDS dataset root directory

Type traits.Str

output_dir
Output directory

Type traits.Str

inspect_outputs
Dictionary of stage outputs with corresponding commands for visual inspection (Initialization: 'Outputs not available')

Type traits.Dict

inspect_outputs_enum
Choice of output to be visually inspected (values='inspect_outputs')

Type traits.Enum

enabled
Stage enabled in the pipeline (Default: True)

Type traits.Bool

config
Instance of stage configuration

Type Instance(HasTraits)

See also:

cmp.stages.preprocessing.preprocessing.PreprocessingStage, `cmp.stages.diffusion.diffusion.DiffusionStage`, *cmp.stages.registration.registration.RegistrationStage*, *cmp.stages.connectome.connectome.ConnectomeStage*, *cmp.stages.preprocessing.fmri_preprocessing.PreprocessingStage*, *cmp.stages.functional.functionalMRI.FunctionalMRIStage*, *cmp.stages.connectome.fmri_connectome.ConnectomeStage*

enabled = True

inspect_outputs = ['Outputs not available']

is_running()

Return the number of unfinished files in the stage.

Returns **nb_of_unfinished_files** – Number of unfinished files in the stage

Return type int

GUI modules

cmp.bidsappmanager.gui package

Module that provides the definition of all classes, functions, and variables dedicated to the GUI of Connectome Mapper 3.

Submodules

cmp.bidsappmanager.gui.bidsapp module

cmp.bidsappmanager.gui.config module

cmp.bidsappmanager.gui.globals module

Modules that defines multiple variables and functions used by the different windows of the GUI.

`cmp.bidsappmanager.gui.globals.`**get_icon**(*icon_fname*)

Return an instance of `ImageResource` or None is there is not graphical backend.

Parameters **icon_fname** (*string*) – Filename to an icon image

Returns **icon** – Return the full path to the icon file or None is there is not graphical backend.

Return type string

cmp.bidsappmanager.gui.handlers module

cmp.bidsappmanager.gui.principal module

cmp.bidsappmanager.gui.qc module

cmp.bidsappmanager.gui.traits module

Module that defines traits-based classes for Connectome Mapper 3 BIDS App Interface TraitsUI View.

class `cmp.bidsappmanager.gui.traits.`**MultiSelectAdapter**

Bases: `traitsui.tabular_adapter.TabularAdapter`

This adapter is used by left and right tables for selection of subject to be processed.

cmp.bidsappmanager.project module

cmp.bidsappmanager.pipelines.anatomical package

Submodules

cmp.bidsappmanager.pipelines.anatomical.anatomical module

Anatomical pipeline UI Class definition.

class `cmp.bidsappmanager.pipelines.anatomical.anatomical.`**`AnatomicalPipelineUI`**(*project_info*)

Bases: *`cmp.pipelines.anatomical.anatomical.AnatomicalPipeline`*

Class that extends the *`AnatomicalPipeline`* with graphical components.

`segmentation`

Button to open the window for configuration or quality inspection of the segmentation stage depending on the `view_mode`

Type traits.ui.Button

`parcellation`

Button to open the window for configuration or quality inspection of the segmentation stage depending on the `view_mode`

Type traits.ui.Button

`view_mode`

Variable used to control the display of either (1) the configuration or (2) the quality inspection of stage of the pipeline

Type ['config_view', 'inspect_outputs_view']

`pipeline_group`

Panel defining the layout of the buttons of the stages with corresponding images

Type traitsUI panel

`traits_view`

QtView that includes the `pipeline_group` panel

Type QtView

See also:

`cmp.pipelines.anatomical.anatomical.AnatomicalPipeline`

`check_input`(*layout*)

Method that checks if inputs of the anatomical pipeline are available in the datasets.

Parameters **`layout`** (*`bids.BIDSLayout`*) – BIDSLayout object used to query

Returns **valid_inputs** – True in all inputs of the anatomical pipeline are available

Return type bool

`check_output`()

Method that checks if outputs of the anatomical pipeline are available.

Returns

- **valid_output** (*bool*) – True is all outputs are found
- **error_message** (*string*) – Message in case there is an error

cmp.bidsappmanager.pipelines.diffusion package

Submodules

cmp.bidsappmanager.pipelines.diffusion.diffusion module

cmp.bidsappmanager.pipelines.functional package

Submodules

cmp.bidsappmanager.pipelines.functional.eeg module

EEG pipeline UI Class definition.

class `cmp.bidsappmanager.pipelines.functional.eeg.`**`EEGPipelineUI`**(*project_info*)

Bases: *`cmp.pipelines.functional.eeg.EEGPipeline`*

Class that extends the *`EEGPipeline`* with graphical components.

`preprocessing`

Button to open the window for configuration or quality inspection of the preprocessing stage depending on the `view_mode`

Type traits.ui.Button

`sourceimaging`

Button to open the window for configuration or quality inspection of the source imaging stage depending on the `view_mode`

Type traits.ui.Button

`connectome`

Button to open the window for configuration or quality inspection of the connectome stage depending on the `view_mode`

Type traits.ui.Button

`view_mode`

Variable used to control the display of either (1) the configuration or (2) the quality inspection of stage of the pipeline

Type ['config_view', 'inspect_outputs_view']

`pipeline_group`

Panel defining the layout of the buttons of the stages with corresponding images

Type traitsUI panel

`traits_view`

QtView that includes the `pipeline_group` panel

Type QtView

See also:

`cmp.pipelines.functional.eeg.EEGPipeline`

cmp.bidsappmanager.pipelines.functional.fMRI module

Functional pipeline UI Class definition.

class cmp.bidsappmanager.pipelines.functional.fMRI.**fMRIPipelineUI**(*project_info*)

Bases: *cmp.pipelines.functional.fMRI.fMRIPipeline*

Class that extends the *fMRIPipeline* with graphical components.

preprocessing

Button to open the window for configuration or quality inspection of the preprocessing stage depending on the `view_mode`

Type traits.ui.Button

registration

Button to open the window for configuration or quality inspection of the registration stage depending on the `view_mode`

Type traits.ui.Button

functionalMRI

Button to open the window for configuration or quality inspection of the extra preprocessing stage stage depending on the `view_mode`

Type traits.ui.Button

connectome

Button to open the window for configuration or quality inspection of the connectome stage depending on the `view_mode`

Type traits.ui.Button

view_mode

Variable used to control the display of either (1) the configuration or (2) the quality inspection of stage of the pipeline

Type ['config_view', 'inspect_outputs_view']

pipeline_group

Panel defining the layout of the buttons of the stages with corresponding images

Type traitsUI panel

traits_view

QtView that includes the `pipeline_group` panel

Type QtView

See also:

cmp.pipelines.functional.fMRI.fMRIPipeline

check_input(*layout*, *gui=True*)

Method that checks if inputs of the fMRI pipeline are available in the datasets.

Parameters

- **layout** (*bids.BIDSLayout*) – BIDSLayout object used to query
- **gui** (*bool*) – If True, display message in GUI

Returns **valid_inputs** – True in all inputs of the fMRI pipeline are available

Return type bool

cmp.bidsappmanager.stages package

Subpackages

cmp.bidsappmanager.stages.connectome package

Submodules

cmp.bidsappmanager.stages.connectome.connectome module

Definition of structural connectome config and stage UI classes.

class `cmp.bidsappmanager.stages.connectome.connectome.`**`ConnectomeConfigUI`**

Bases: *cmp.stages.connectome.connectome.ConnectomeConfig*

Class that extends the `ConnectomeConfig` with graphical components.

`output_types`

A list of `output_types`. Valid `output_types` are 'gpickle', 'mat', 'cff', 'graphml'

Type list of string

`connectivity_metrics`

A list of connectivity metrics to stored. Valid `connectivity_metrics` are 'Fiber number', 'Fiber length', 'Fiber density', 'Fiber proportion', 'Normalized fiber density', 'ADC', 'gFA'

Type list of string

`traits_view`

TraitsUI view that displays the Attributes of this class

Type traits.ui.View

See also:

cmp.stages.connectome.connectome.ConnectomeConfig

class `cmp.bidsappmanager.stages.connectome.connectome.`**`ConnectomeStageUI`**(*bids_dir*, *output_dir*)

Bases: *cmp.stages.connectome.connectome.ConnectomeStage*

Class that extends the `ConnectomeStage` with graphical components.

`log_visualization`

If True, display with a log transformation

Type traits.Bool

`circular_layout`

If True, display the connectivity matrix using a circular layout

Type traits.Bool

`inspect_output_button`

Button that displays the selected connectivity matrix in the graphical component for quality inspection

Type traits.ui.Button

`inspect_outputs_view`

TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view
TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

cmp.stages.connectome.connectome.ConnectomeStage

cmp.bidsappmanager.stages.connectome.eeg_connectome module

Definition of EEG connectome config and stage UI classes.

class cmp.bidsappmanager.stages.connectome.eeg_connectome.**EEGConnectomeConfigUI**
Bases: *cmp.stages.connectome.eeg_connectome.EEGConnectomeConfig*

Class that extends the *cmp.stages.connectome.eeg_connectome.EEGConnectomeConfig* with graphical components.

output_types
A list of output_types. Valid output_types are 'gpickle', 'mat', 'cff', 'graphml'

Type list of string

connectivity_metrics
A list of time/frequency connectivity metrics to stored. Valid connectivity_metrics are 'coh', 'cohy', 'imcoh', 'plv', 'ciplv', 'ppc', 'pli', 'wpli', and 'wpli2_debiased'

Type list of string

traits_view
TraitsUI view that displays the Attributes of this class

Type traits.ui.View

See also:

cmp.stages.connectome.eeg_connectome.EEGConnectomeConfig

class cmp.bidsappmanager.stages.connectome.eeg_connectome.**EEGConnectomeStageUI**(*subject*, *session*, *bids_dir*, *output_dir*)
Bases: *cmp.stages.connectome.eeg_connectome.EEGConnectomeStage*

Class that extends the *cmp.stages.connectome.eeg_connectome.EEGConnectomeStage* with graphical components.

log_visualization
Log visualization that might be obsolete as this has been detached after creation of the bidsappmanager (Default: True)

Type traits.Bool

circular_layout
Visualization of the connectivity matrix using a circular layout that might be obsolete as this has been detached after creation of the bidsappmanager (Default: False)

Type traits.Bool

inspect_output_button
Button that displays the selected connectivity matrix in the graphical component for quality inspection

Type traits.ui.Button

inspect_outputs_view
TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view
TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

cmp.stages.connectome.eeg_connectome.EEGConnectomeStage

cmp.bidsappmanager.stages.connectome.fmri_connectome module

Definition of functional connectome config and stage UI classes.

class cmp.bidsappmanager.stages.connectome.fmri_connectome.**ConnectomeConfigUI**
Bases: *cmp.stages.connectome.fmri_connectome.ConnectomeConfig*

Class that extends the `ConnectomeConfig` with graphical components.

output_types
A list of `output_types`. Valid `output_types` are 'gpickle', 'mat', 'cff', 'graphml'

Type list of string

traits_view
TraitsUI view that displays the Attributes of this class

Type traits.ui.View

See also:

cmp.stages.connectome.fmri_connectome.ConnectomeConfig

class cmp.bidsappmanager.stages.connectome.fmri_connectome.**ConnectomeStageUI**(*bids_dir*, *output_dir*)
Bases: *cmp.stages.connectome.fmri_connectome.ConnectomeStage*

Class that extends the `ConnectomeStage` with graphical components.

log_visualization
If True, display with a log transformation

Type traits.Bool

circular_layout
If True, display the connectivity matrix using a circular layout

Type traits.Bool

inspect_output_button
Button that displays the selected connectivity matrix in the graphical component for quality inspection

Type traits.ui.Button

inspect_outputs_view
TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view
TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

cmp.stages.connectome.fmri_connectome.ConnectomeStage

cmp.bidsappmanager.stages.diffusion package

Submodules

cmp.bidsappmanager.stages.diffusion.diffusion module

cmp.bidsappmanager.stages.diffusion.reconstruction module

cmp.bidsappmanager.stages.diffusion.tracking module

cmp.bidsappmanager.stages.eeg package

Submodules

cmp.bidsappmanager.stages.eeg.esi module

Definition of EEG Source Imaging config and stage UI classes.

`class cmp.bidsappmanager.stages.eeg.esi.`**EEGSourceImagingConfigUI**
Bases: *cmp.stages.eeg.esi.EEGSourceImagingConfig*

Class that extends the *cmp.stages.eeg.esi.EEGSourceImagingConfig* with graphical components.

traits_view
TraitsUI view that displays the attributes of this class, e.g. the parameters for the stage

Type traits.ui.View

See also:

cmp.stages.eeg.esi.EEGSourceImagingConfig

`class cmp.bidsappmanager.stages.eeg.esi.`**EEGSourceImagingStageUI**(*subject*, *session*, *bids_dir*, *output_dir*)
Bases: *cmp.stages.eeg.esi.EEGSourceImagingStage*

Class that extends the *cmp.stages.eeg.esi.EEGSourceImagingStage* with graphical components.

inspect_output_button
Button that displays the selected output in an appropriate viewer (present only in the window for quality inspection)

Type traits.ui.Button

inspect_outputs_view
TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view
TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

cmp.stages.eeg.esi.EEGSourceImagingStage

cmp.bidsappmanager.stages.eeg.preprocessing module

Definition of EEG preprocessing config and stage UI classes.

class cmp.bidsappmanager.stages.eeg.preprocessing.**EEGPreprocessingConfigUI**
Bases: *cmp.stages.eeg.preprocessing.EEGPreprocessingConfig*

Class that extends the *cmp.stages.eeg.preprocessing.EEGPreprocessingConfig* with graphical components.

traits_view
TraitsUI view that displays the attributes of this class, e.g. the parameters for the stage

Type traits.ui.View

See also:

cmp.stages.eeg.preprocessing.EEGPreprocessingConfig

class cmp.bidsappmanager.stages.eeg.preprocessing.**EEGPreprocessingStageUI**(*subject*, *session*, *bids_dir*, *output_dir*)
Bases: *cmp.stages.eeg.preprocessing.EEGPreprocessingStage*

Class that extends the *cmp.stages.eeg.preprocessing.EEGPreprocessingStage* with graphical components.

inspect_output_button
Button that displays the selected output in an appropriate viewer (present only in the window for quality inspection)

Type traits.ui.Button

inspect_outputs_view
TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view
TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

cmp.stages.eeg.preprocessing.EEGPreprocessingStage

cmp.bidsappmanager.stages.functional package

Submodules

cmp.bidsappmanager.stages.functional.functionalMRI module

Definition of extra preprocessing of functional MRI (post-registration) config and stage UI classes.

class `cmp.bidsappmanager.stages.functional.functionalMRI.`**`FunctionalMRIConfigUI`**

Bases: *`cmp.stages.functional.functionalMRI.FunctionalMRIConfig`*

Class that extends the `FunctionalMRIConfig` with graphical components.

`traits_view`

TraitsUI view that displays the attributes of this class, e.g. the parameters for the stage

Type traits.ui.View

See also:

`cmp.stages.functional.functionalMRI.FunctionalMRIConfig`

class `cmp.bidsappmanager.stages.functional.functionalMRI.`**`FunctionalMRIStageUI`**(*bids_dir*, *output_dir*)

Bases: *`cmp.stages.functional.functionalMRI.FunctionalMRIStage`*

Class that extends the `FunctionalMRIStage` with graphical components.

`inspect_output_button`

Button that displays the selected output in an appropriate viewer (present only in the window for quality inspection)

Type traits.ui.Button

`inspect_outputs_view`

TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

`config_view`

TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

`cmp.stages.functional.functionalMRI.FunctionalMRIStage`

cmp.bidsappmanager.stages.parcellation package

Submodules

cmp.bidsappmanager.stages.parcellation.parcellation module

Definition of parcellation config and stage UI classes.

class `cmp.bidsappmanager.stages.parcellation.parcellation.`**`ParcellationConfigUI`**

Bases: *`cmp.stages.parcellation.parcellation.ParcellationConfig`*

Class that extends the `ParcellationConfig` with graphical components.

custom_parcellation_view
VGroup that displays the different parts of a custom BIDS parcellation file

Type traits.ui.View

traits_view
TraitsUI view that displays the attributes of this class, e.g. the parameters for the stage

Type traits.ui.View

See also:

cmp.stages.parcellation.parcellation.ParcellationConfig

class cmp.bidsappmanager.stages.parcellation.parcellation.**ParcellationStageUI**(*pipeline_mode*, *subject*, *session*, *bids_dir*, *output_dir*)

Bases: *cmp.stages.parcellation.parcellation.ParcellationStage*

Class that extends the `ParcellationStage` with graphical components.

inspect_output_button
Button that displays the selected output in an appropriate viewer (present only in the window for quality inspection)

Type traits.ui.Button

inspect_outputs_view
TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view
TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

cmp.stages.parcellation.parcellation.ParcellationStage

cmp.bidsappmanager.stages.preprocessing package

Submodules

cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing module

Definition of fMRI preprocessing config and stage UI classes.

class cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing.**PreprocessingConfigUI**

Bases: *cmp.stages.preprocessing.fmri_preprocessing.PreprocessingConfig*

Class that extends the (functional) `PreprocessingConfig` with graphical components.

traits_view
TraitsUI view that displays the attributes of this class, e.g. the parameters for the stage

Type traits.ui.View

See also:

cmp.stages.preprocessing.fmri_preprocessing.PreprocessingConfig

class cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing.**PreprocessingStageUI**(*bids_dir*, *output_dir*)

Bases: *cmp.stages.preprocessing.fmri_preprocessing.PreprocessingStage*

Class that extends the (functional) `PreprocessingStage` with graphical components.

inspect_output_button

Button that displays the selected output in an appropriate viewer (present only in the window for quality inspection)

Type traits.ui.Button

inspect_outputs_view

TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view

TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

cmp.stages.preprocessing.fmri_preprocessing.PreprocessingStage

cmp.bidsappmanager.stages.preprocessing.preprocessing module

Definition of diffusion preprocessing config and stage UI classes.

class cmp.bidsappmanager.stages.preprocessing.preprocessing.**PreprocessingConfigUI**

Bases: *cmp.stages.preprocessing.preprocessing.PreprocessingConfig*

Class that extends the (diffusion) `PreprocessingConfig` with graphical components.

traits_view

TraitsUI view that displays the attributes of this class, e.g. the parameters for the stage

Type traits.ui.View

See also:

cmp.stages.preprocessing.preprocessing.PreprocessingConfig

class cmp.bidsappmanager.stages.preprocessing.preprocessing.**PreprocessingStageUI**(*bids_dir*, *output_dir*)

Bases: *cmp.stages.preprocessing.preprocessing.PreprocessingStage*

Class that extends the (diffusion) `PreprocessingStage` with graphical components.

inspect_output_button

Button that displays the selected output in an appropriate viewer (present only in the window for quality inspection)

Type traits.ui.Button

inspect_outputs_view

TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view
TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

cmp.stages.preprocessing.preprocessing.PreprocessingStage

cmp.bidsappmanager.stages.registration package

Submodules

cmp.bidsappmanager.stages.registration.registration module

Definition of registration config and stage UI classes.

class cmp.bidsappmanager.stages.registration.registration.**RegistrationConfigUI**
Bases: *cmp.stages.registration.registration.RegistrationConfig*

Class that extends the `RegistrationConfig` with graphical components.

traits_view
TraitsUI view that displays the attributes of this class, e.g. the parameters for the stage

Type traits.ui.View

See also:

cmp.stages.registration.registration.RegistrationConfig

class cmp.bidsappmanager.stages.registration.registration.**RegistrationStageUI**(*pipeline_mode*, *fs_subjects_dir=None*, *fs_subject_id=None*, *bids_dir=''*, *output_dir=''*)
Bases: *cmp.stages.registration.registration.RegistrationStage*

Class that extends the `RegistrationStage` with graphical components.

inspect_output_button
Button that displays the selected output in an appropriate viewer (present only in the window for quality inspection)

Type traits.ui.Button

inspect_outputs_view
TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view
TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

cmp.stages.registration.registration.RegistrationStage

cmp.bidsappmanager.stages.segmentation package

Submodules

cmp.bidsappmanager.stages.segmentation.segmentation module

Definition of segmentation config and stage UI classes.

class `cmp.bidsappmanager.stages.segmentation.segmentation.`**`SegmentationConfigUI`**

Bases: *`cmp.stages.segmentation.segmentation.SegmentationConfig`*

Class that extends the `SegmentationConfig` with graphical components.

`custom_brainmask_group`

VGroup that displays the different parts of a custom BIDS brain mask file

Type traits.ui.VGroup

`custom_gm_mask_group`

VGroup that displays the different parts of a custom BIDS gray matter mask file

Type traits.ui.VGroup

`custom_wm_mask_group`

VGroup that displays the different parts of a custom BIDS white matter mask file

Type traits.ui.VGroup

`custom_csf_mask_group`

VGroup that displays the different parts of a custom BIDS CSF mask file

Type traits.ui.VGroup

`custom_aparcaseg_group`

VGroup that displays the different parts of a custom BIDS-formatted Freesurfer's aparc+aseg file

Type traits.ui.VGroup

`traits_view`

TraitsUI view that displays the attributes of this class, e.g. the parameters for the stage

Type traits.ui.View

See also:

`cmp.stages.segmentation.segmentation.SegmentationConfig`

class `cmp.bidsappmanager.stages.segmentation.segmentation.`**`SegmentationStageUI`**(*subject*, *session*, *bids_dir*, *output_dir*)

Bases: *`cmp.stages.segmentation.segmentation.SegmentationStage`*

Class that extends the `SegmentationStage` with graphical components.

`inspect_output_button`

Button that displays the selected output in an appropriate viewer (present only in the window for quality inspection)

Type traits.ui.Button

`inspect_outputs_view`

TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view
TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

cmp.stages.segmentation.segmentation.SegmentationStage

5.6.2 cmtklib package

Subpackages

cmtklib.bids package

Submodules

cmtklib.bids.io module

This module provides classes to handle custom BIDS derivatives file input.

class `cmtklib.bids.io.`**CustomAparcAsegBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom BIDS-formatted Freesurfer aparc+aseg file in the form `sub-<label>_desc-aparcaseg_dseg.nii.gz`.

class `cmtklib.bids.io.`**CustomBIDSFile**(*p_toolbox_derivatives_dir='', p_datatype='', p_suffix='', p_extension='', p_acquisition='', p_rec='', p_atlas='', p_res='', p_label='', p_desc='', p_task=''*)
Bases: `traits.has_traits.HasTraits`

Base class used to represent a BIDS-formatted file inside a custom BIDS derivatives directory.

toolbox_derivatives_dir
Toolbox folder name in the `derivatives/` of the BIDS dataset

Type Str

datatype
BIDS data type

Type Enum([“anat”, “dwi”, “func”, “eeg”])

suffix
Filename suffix e.g. `sub-01_T1w.nii.gz` has suffix `T1w`

Type Str

acquisition
Label used in `_acq-<label>_`

Type Str

task
Label used in `_task-<label>_`

Type Str

rec
Label used in `_rec-<label>_`

Type Str

res
Label used in `_res-<label>_`

Type Str

extension
File extension

Type Str

atlas
Label used in `_atlas-<label>_`

Type Str

label
Label used in `_label-<label>_`

Type Str

desc
Label used in `_desc-<label>_`

Type Str

get_filename(*subject*, *session=None*, *debug=False*)
Return the filename path with extension of the represented BIDS file.

Parameters

- **subject** (*str*) – Subject filename entity e.g. "sub-01"
- **session** (*str*) – Session filename entity e.g. "ses-01" if applicable (Default: None)
- **debug** (*bool*) – Debug mode (Extra output messages) if `True`

get_filename_path(*base_dir*, *subject*, *session=None*, *debug=False*)
Return the filename path without extension of the represented BIDS file.

Parameters

- **base_dir** (*str*) – BIDS root directory or `derivatives/` directory in BIDS root directory
- **subject** (*str*) – Subject filename entity e.g. "sub-01"
- **session** (*str*) – Session filename entity e.g. "ses-01" if applicable (Default: None)
- **debug** (*bool*) – Debug mode (Extra output messages) if `True`

get_query_dict()
Return the dictionary to be passed to `BIDSDataGrabber` to query a list of files.

get_toolbox_derivatives_dir()
Return the value of `custom_derivatives_dir` attribute.

class cmtklib.bids.io.**CustomBrainMaskBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom brain mask in the form `sub-<label>_desc-brain_mask.nii.gz`.

class cmtklib.bids.io.**CustomCSFMaskBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom CSF mask in the form `sub-<label>_label-CSF_dseg.nii.gz`.

class cmtklib.bids.io.**CustomEEGCartoolElectrodesBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom BIDS-formatted electrode file produced by Cartool, in the form `sub-<label>_eeg.xyz`.

class cmtklib.bids.io.**CustomEEGCartoolInvSolBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom BIDS-formatted inverse solution file produced by Cartool in the form `sub-<label>_eeg.[LAURA|LORETA].is`.

class cmtklib.bids.io.**CustomEEGCartoolMapSpiRoisBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom BIDS-formatted spi / rois mapping file in the form `sub-<label>_eeg.pickle.rois`.

class cmtklib.bids.io.**CustomEEGCartoolSpiBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom BIDS-formatted Source Point Irregularly spaced file produced by Cartool, in the form `sub-<label>_eeg.spi`.

class cmtklib.bids.io.**CustomEEGElectrodesBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom BIDS-formatted EEG electrodes file in the form `sub-<label>_task-<label>_electrodes.tsv`.

class cmtklib.bids.io.**CustomEEGEpochsBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom BIDS-formatted EEG Epochs file in .set or .fif format.

class cmtklib.bids.io.**CustomEEGEventsBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom BIDS-formatted EEG task events file in the form `sub-<label>_task-<label>_events.tsv`.

extract_event_ids_from_json_sidecar(*base_dir*, *subject*, *session=None*, *debug=False*)

class cmtklib.bids.io.**CustomEEGMNETransformBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom BIDS-formatted electrode transform file in the form `sub-<label>_trans.fif`.

class cmtklib.bids.io.**CustomEEGPreprocBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom BIDS-formatted preprocessed EEG file in the form `sub-<label>_task-<label>_desc-preproc_eeg.[set|fif]`.

class cmtklib.bids.io.**CustomGMMaskBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom gray-matter mask in the form `sub-<label>_label-GM_dseg.nii.gz`.

class cmtklib.bids.io.**CustomParcellationBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom parcellation files in the form `sub-<label>_atlas-<label>[_res-<label>]_dseg.nii.gz`.

get_nb_of_regions(*bids_dir*, *subject*, *session=None*, *debug=False*)
Return the number of regions by reading its associated TSV side car file describing the nodes.

Parameters

- **bids_dir** (*str*) – BIDS root directory
- **subject** (*str*) – Subject filename entity e.g. "sub-01"
- **session** (*str*) – Session filename entity e.g. "ses-01" if applicable (Default: None)
- **debug** (*bool*) – Debug mode (Extra output messages) if True

class cmtklib.bids.io.**CustomWMMaskBIDSFile**
Bases: *cmtklib.bids.io.CustomBIDSFile*

Represent a custom white-matter mask in the form `sub-<label>_label-WM_dseg.nii.gz`.

cmtklib.bids.network module

This module provides functions to handle connectome networks / graphs generated by CMP3.

cmtklib.bids.network.**load_graphs**(*output_dir*, *subjects*, *parcellation_scheme*, *weight*)
Return a dictionary of connectivity matrices (graph adjacency matrices).

Still in development

Parameters

- **output_dir** (*string*) – Output/derivatives directory
- **subjects** (*list*) – List of subject
- **parcellation_scheme** (*['NativeFreesurfer', 'Lausanne2018', 'Custom']*) – Parcellation scheme
- **weight** (*['number_of_fibers','fiber_density',...]*) – Edge metric to extract from the graph

Returns **connmats** – Dictionary of connectivity matrices

Return type dict

cmtklib.bids.utils module

This module provides CMTK Utility functions to handle BIDS datasets.

CreateBIDSStandardParcellationLabelIndexMappingFile

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Creates the BIDS standard generic label-index mapping file that describes parcellation nodes.

roi_colorlut [a pathlike object or string representing an existing file] Path to FreesurferColorLUT.txt file that describes the RGB color of the graph nodes for a given parcellation.

roi_graphml [a pathlike object or string representing an existing file] Path to graphml file that describes graph nodes for a given parcellation.

verbose [a boolean] Verbose mode.

roi_bids_tsv [a pathlike object or string representing a file] Output BIDS standard generic label-index mapping file that describes parcellation nodes.

CreateCMPParcellationNodeDescriptionFilesFromBIDSFile

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Creates CMP graphml and FreeSurfer colorLUT files that describe parcellation nodes from the BIDS TSV file

roi_bids_tsv [a pathlike object or string representing an existing file] Output BIDS standard generic label-index mapping file that describes parcellation nodes.

roi_colorlut [a pathlike object or string representing a file] Path to FreesurferColorLUT.txt file that describes the RGB color of the graph nodes for a given parcellation.

roi_graphml [a pathlike object or string representing a file] Path to graphml file that describes graph nodes for a given parcellation.

CreateMultipleCMPParcellationNodeDescriptionFilesFromBIDSFile

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Creates CMP graphml and FreeSurfer colorLUT files describing parcellation nodes from a list of BIDS TSV files

roi_bids_tsvs : a list of items which are a pathlike object or string representing an existing file

roi_colorluts : a list of items which are a pathlike object or string representing a file roi_graphmls : a list of items which are a pathlike object or string representing a file

`cmtklib.bids.utils.`**`get_native_space_files`**(*filepathlist*)
Return a list of files without `_space-<label>_` in the filename.

`cmtklib.bids.utils.`**`get_native_space_no_desc_files`**(*filepathlist*)
Return a list of files without `_space-<label>_` and `_desc-<label>_` in the filename.

`cmtklib.bids.utils.`**`get_native_space_tsv_sidecar_files`**(*filepathlist*)
Return path to tsv sidecar file of a list of niftis (`nii.gz`) without `_space-<label>_` in their filename.

`cmtklib.bids.utils.`**`write_derivative_description`**`(`*bids_dir*, *deriv_dir*, *pipeline_name*`)`

Write a dataset_description.json in each type of CMP derivatives.

Parameters

- **`bids_dir`** (*`string`*) – BIDS root directory
- **`deriv_dir`** (*`string`*) – Output/derivatives directory
- **`pipeline_name`** (*`string`*) – Type of derivatives (`['cmp-<version>', 'freesurfer-<version>', 'nipype-<version>']`)

cmtklib.interfaces package

Submodules

cmtklib.interfaces.afni module

The AFNI module provides Nipype interfaces for the AFNI toolbox missing in nipype or modified.

Bandpass

Link to code

Bases: nipype.interfaces.afni.base.AFNICommand

Wrapped executable: `3dBandpass`.

Program to lowpass and/or highpass each voxel time series in a dataset.

Calls the `3dBandpass` tool from AFNI, offering more/different options than Fourier.

For complete details, see the 3dBandpass Documentation.

Examples

```
>>> from nipype.interfaces import afni as afni
>>> from nipype.testing import  example_data
>>> bandpass = afni.Bandpass()
>>> bandpass.inputs.in_file = example_data('functional.nii')
>>> bandpass.inputs.highpass = 0.005
>>> bandpass.inputs.lowpass = 0.1
>>> res = bandpass.run()
```

highpass [a float] Highpass. Maps to a command-line argument: `%f` (position: -3).

in_file [a pathlike object or string representing an existing file] Input file to 3dBandpass. Maps to a command-line argument: `%s` (position: -1).

lowpass [a float] Lowpass. Maps to a command-line argument: `%f` (position: -2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

automask [a boolean] Create a mask from the input dataset. Maps to a command-line argument: `-automask`.

blur [a float] Blur (inside the mask only) with a filter width (FWHM) of 'fff' millimeters. Maps to a command-line argument: `-blur %f`.

despike [a boolean] Despike each time series before other processing. Hopefully, you don't actually need to do this, which is why it is optional. Maps to a command-line argument: `-despike`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

localPV [a float] Replace each vector by the local Principal Vector (AKA first singular vector) from a neighborhood of radius 'rrr' millimiters. Note that the PV time series is L2 normalized. This option is mostly for Bob Cox to have fun with. Maps to a command-line argument: `-localPV %f`.

mask [a pathlike object or string representing an existing file] Mask file. Maps to a command-line argument: `-mask %s` (position: 2).

nfft [an integer] Set the FFT length [must be a legal value]. Maps to a command-line argument: `-nfft %d`.

no_detrend [a boolean] Skip the quadratic detrending of the input that occurs before the FFT-based bandpassing. ++ You would only want to do this if the dataset had been detrended already in some other program. Maps to a command-line argument: `-nodetrend`.

normalize [a boolean] Make all output time series have L2 norm = 1 ++ i.e., sum of squares = 1. Maps to a command-line argument: `-norm`.

notrans [a boolean] Don't check for initial positive transients in the data: The test is a little slow, so skipping it is OK, if you KNOW the data time series are transient-free. Maps to a command-line argument: `-notrans`.

num_threads [an integer] Set number of threads. (Nipype **default** value: `1`)

orthogonalize_dset [a pathlike object or string representing an existing file] Orthogonalize each voxel to the corresponding voxel time series in dataset 'fset', which must have the same spatial and temporal grid structure as the main input dataset. At present, only one '-dsort' option is allowed. Maps to a command-line argument: `-dsort %s`.

orthogonalize_file [a list of items which are a pathlike object or string representing an existing file] Also orthogonalize input to columns in f.1D Multiple '-ort' options are allowed. Maps to a command-line argument: `-ort %s`.

out_file [a pathlike object or string representing a file] Output file from 3dBandpass. Maps to a command-line argument: `-prefix %s` (position: 1).

outputtype ['NIFTI' or 'AFNI' or 'NIFTI_GZ'] AFNI output filetype.

tr [a float] Set time step (TR) in sec [default=from dataset header]. Maps to a command-line argument: `-dt %f`.

out_file [a pathlike object or string representing an existing file] Output file.

Despike

Link to code

Bases: nipype.interfaces.afni.base.AFNICommand

Wrapped executable: `3dDespike`.

Removes 'spikes' from the 3D+time input dataset.

It calls the `3dDespike` tool from AFNI.

For complete details, see the 3dDespike Documentation.

Examples

```
>>> from nipype.interfaces import afni
>>> despike = afni.Despike()
>>> despike.inputs.in_file = 'functional.nii'
>>> res = despike.run()
```

in_file [a pathlike object or string representing an existing file] Input file to 3dDespike. Maps to a command-line argument: `%s` (position: -1).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

num_threads [an integer] Set number of threads. (Nipype **default** value: `1`)

out_file [a pathlike object or string representing a file] Output image file name. Maps to a command-line argument: `-prefix %s`.

outputtype ['NIFTI' or 'AFNI' or 'NIFTI_GZ'] AFNI output filetype.

out_file [a pathlike object or string representing an existing file] Output file.

cmtklib.interfaces.ants module

The ANTs module provides Nipype interfaces for the ANTs registration toolbox missing in nipype or modified.

MultipleANTsApplyTransforms

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Apply linear and deformable transforms estimated by ANTS to a list of images.

It calls the `antsApplyTransform` on a series of images.

Examples

```
>>> apply_tf = MultipleANTsApplyTransforms()
>>> apply_tf.inputs.input_images = ['/path/to/sub-01_atlas-L2018_desc-scale1_
↪dseg.nii.gz',
>>>                                 '/path/to/sub-01_atlas-L2018_desc-scale2_
↪dseg.nii.gz',
>>>                                 '/path/to/sub-01_atlas-L2018_desc-scale3_
↪dseg.nii.gz',
>>>                                 '/path/to/sub-01_atlas-L2018_desc-scale4_
↪dseg.nii.gz',
>>>                                 '/path/to/sub-01_atlas-L2018_desc-scale5_
↪dseg.nii.gz']
>>> apply_tf.inputs.transforms = ['/path/to/final1Warp.nii.gz',
>>>                               '/path/to/final0GenericAffine.mat']
>>> apply_tf.inputs.reference_image = File(mandatory=True, exists=True)
>>> apply_tf.inputs.interpolation = 'NearestNeighbor'
>>> apply_tf.inputs.default_value = 0.0
>>> apply_tf.inputs.out_postfix = "_transformed"
>>> apply_tf.run()
```

reference_image : a string or os.PathLike object referring to an existing file transforms : a list of items which are a string or os.PathLike object referring to an existing file

Transform files: will be applied in reverse order. For example, the last specified transform will be applied first.

default_value : a float input_images : a list of items which are a string or os.PathLike object referring to an existing file interpolation : 'Linear' or 'NearestNeighbor' or 'CosineWindowedSinc' or 'WelchWindowedSinc' or 'HammingWindowedSinc' or 'LanczosWindowedSinc' or 'MultiLabel' or 'Gaussian' or 'BSpline'

(Nipype **default** value: `Linear`)

out_postfix [a string] (Nipype **default** value: `_transformed`)

output_images : a list of items which are a string or os.PathLike object

cmtklib.interfaces.dipy module

cmtklib.interfaces.freesurfer module

The FreeSurfer module provides Nipype interfaces for Freesurfer tools missing in nipype or modified.

BBRegister

Link to code

Bases: nipype.interfaces.freesurfer.base.FSCommand

Wrapped executable: `bbregister`.

Use FreeSurfer bbregister to register a volume to the Freesurfer anatomical.

This program performs within-subject, cross-modal registration using a boundary-based cost function. The registration is constrained to be 6 DOF (rigid).

It is required that you have an anatomical scan of the subject that has already been `recon-all-ed` using freesurfer.

Examples

```
>>> from cmtklib.interfaces.freesurfer import BBRegister
>>> bbreg = BBRegister(subject_id='me',
>>>                    source_file='structural.nii',
>>>                    init='header',
>>>                    contrast_type='t2')
>>> bbreg.run()
```

contrast_type ['t1' or 't2' or 'dti'] Contrast type of image. Maps to a command-line argument: `--%s`.

init ['spm' or 'fsl' or 'header'] Initialize registration spm, fsl, header. Maps to a command-line argument: `--init-%s`. Mutually **exclusive** with inputs: `init_reg_file`.

init_reg_file [a pathlike object or string representing an existing file] Existing registration file. Mutually **exclusive** with inputs: `init`.

source_file [a pathlike object or string representing a file] Source file to be registered. Maps to a command-line argument: `--mov %s`.

subject_id [a string] Freesurfer subject id. Maps to a command-line argument: `--s %s`.

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

epi_mask [a boolean] Mask out B0 regions in stages 1 and 2. Maps to a command-line argument: `--epi-mask`.

intermediate_file [a pathlike object or string representing an existing file] Intermediate image, e.g. in case of partial FOV. Maps to a command-line argument: `--int %s`.

out_fsl_file [a boolean or a pathlike object or string representing a file] Write the transformation matrix in FSL FLIRT format. Maps to a command-line argument: `--fslmat %s`.

out_reg_file [a pathlike object or string representing a file] Output registration file. Maps to a command-line argument: `--reg %s`.

reg_frame [an integer] 0-based frame index for 4D source file. Maps to a command-line argument: `--frame %d`. Mutually **exclusive** with inputs: `reg_middle_frame`.

reg_middle_frame [a boolean] Register middle frame of 4D source file. Maps to a command-line argument: `--mid-frame`. Mutually **exclusive** with inputs: `reg_frame`.

registered_file [a boolean or a pathlike object or string representing a file] Output warped sourcefile either True or filename. Maps to a command-line argument: `--o %s`.

spm_nifti [a boolean] Force use of nifti rather than analyze with SPM. Maps to a command-line argument: `--spm-nii`.

subjects_dir [a pathlike object or string representing an existing directory] Subjects directory.

min_cost_file [a pathlike object or string representing an existing file] Output registration minimum cost file.

out_fsl_file [a pathlike object or string representing a file] Output FLIRT-style registration file.

out_reg_file [a pathlike object or string representing an existing file] Output registration file.

registered_file [a pathlike object or string representing a file] Registered and resampled source file.

Tkregister2

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `tkregister2`.

Performs image co-registration using Freesurfer `tkregister2`.

Examples

```
>>> from cmtklib.interfaces.freesurfer import Tkregister2
>>> tkreg = Tkregister2()
>>> tkreg.inputs.in_file = 'sub-01_desc-brain_mask.nii.gz'
>>> tkreg.inputs.subject_dir = '/path/to/output_dir/freesurfer/sub-01'
>>> tkreg.inputs.subjects_dir = '/path/to/output_dir/freesurfer'
>>> tkreg.inputs.subject_id = 'sub-01'
>>> tkreg.inputs.regheader = True
>>> tkreg.inputs.in_file = '/path/to/moving_image.nii.gz'
>>> tkreg.inputs.target_file = '/path/to/fixed_image.nii.gz'
>>> tkreg.inputs.fslreg_out = 'motions.par'
>>> tkreg.inputs.noedit = True
>>> tkreg.run()
```

fslreg_out [a string] FSL-Style registration output matrix. Maps to a command-line argument: `--fslregout %s`.

in_file [a pathlike object or string representing a file] Movable volume. Maps to a command-line argument: `--mov %s`.

reg_out [a string] Input/output registration file. Maps to a command-line argument: `--reg %s`.

target_file [a pathlike object or string representing a file] Target volume. Maps to a command-line argument: `--targ %s`.

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

noedit [a boolean] Do not open edit window (exit) - for conversions. Maps to a command-line argument: `--noedit`.

regheader [a boolean] Compute registration from headers. Maps to a command-line argument: `--regheader`.

subject_id [a string] Set subject id. Maps to a command-line argument: `--s %s`.

subjects_dir [a pathlike object or string representing an existing directory] Use dir as SUBJECTS_DIR. Maps to a command-line argument: `--sd %s`.

fslregout_file [a pathlike object or string representing a file] Resulting FSL-Style registration matrix.

regout_file [a pathlike object or string representing a file] Resulting registration file.

copyBrainMaskToFreesurfer

Link to code

Bases: nipype.interfaces.io.IOBase

Copy a custom brain mask in the freesurfer subject `mri/` directory.

It replaces the brainmask files generated by Freesurfer recon-all in order to re-run recon-all with a custom brain mask.

Examples

```
>>> from cmtklib.interfaces.freesurfer import copyBrainMaskToFreesurfer
>>> copy_mask_fs = copyBrainMaskToFreesurfer()
>>> copy_mask_fs.inputs.in_file = 'sub-01_desc-brain_mask.nii.gz'
>>> copy_mask_fs.inputs.subject_dir = '/path/to/output_dir/freesurfer/sub-01'
>>> copy_mask_fs.run()
```

in_file : a pathlike object or string representing an existing file subject_dir : a pathlike object or string representing an existing directory

out_brainmask_file : a pathlike object or string representing an existing file out_brainmaskauto_file : a pathlike object or string representing an existing file

copyFileToFreesurfer

Link to code

Bases: nipype.interfaces.io.IOBase

Copy a file to an output specified.

Note: Not used.

in_file : a pathlike object or string representing an existing file out_file : a pathlike object or string representing a file

out_file : a pathlike object or string representing an existing file

cmtklib.interfaces.fsl module

The FSL module provides Nipype interfaces for FSL functions missing in Nipype or modified.

ApplymultipleWarp

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Apply a deformation field estimated by FSL `fnirt` to a list of images.

Example

```
>>> from cmtklib.interfaces import fsl
>>> apply_warp = fsl.ApplymultipleWarp()
>>> apply_warp.inputs.in_files = ['/path/to/sub-01_atlas-L2018_desc-scale1_
↪dseg.nii.gz',
>>>                               '/path/to/sub-01_atlas-L2018_desc-scale2_
↪dseg.nii.gz',
>>>                               '/path/to/sub-01_atlas-L2018_desc-scale3_
↪dseg.nii.gz',
>>>                               '/path/to/sub-01_atlas-L2018_desc-scale4_
↪dseg.nii.gz',
>>>                               '/path/to/sub-01_atlas-L2018_desc-scale5_
↪dseg.nii.gz']
>>> apply_warp.inputs.field_file = '/path/to/fnirt_deformation.nii.gz'
>>> apply_warp.inputs.ref_file = '/path/to/sub-01_meanBOLD.nii.gz'
>>> apply_warp.run()
```

field_file [a pathlike object or string representing an existing file] Deformation field.

ref_file [a pathlike object or string representing an existing file] Reference image used for target space.

in_files [a list of items which are a pathlike object or string representing an existing file] Files to be registered.

interp ['nn' or 'trilinear' or 'sinc' or 'spline'] Interpolation method. Maps to a command-line argument: `--interp=%s` (position: -2).

out_files [a list of items which are a pathlike object or string representing a file] Warped files.

ApplymultipleXfm

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Apply an XFM transform estimated by FSL `flirt` to a list of images.

Example

```
>>> from cmtklib.interfaces import fsl
>>> apply_xfm = fsl.ApplymultipleXfm
>>> apply_xfm.inputs.in_files = ['/path/to/sub-01_atlas-L2018_desc-scale1_dseg.
↪nii.gz',
>>>                              '/path/to/sub-01_atlas-L2018_desc-scale2_dseg.
↪nii.gz',
>>>                              '/path/to/sub-01_atlas-L2018_desc-scale3_dseg.
↪nii.gz',
>>>                              '/path/to/sub-01_atlas-L2018_desc-scale4_dseg.
↪nii.gz',
>>>                              '/path/to/sub-01_atlas-L2018_desc-scale5_dseg.
↪nii.gz']
>>> apply_xfm.inputs.xfm_file = '/path/to/flirt_transform.xfm'
>>> apply_xfm.inputs.reference = '/path/to/sub-01_meanBOLD.nii.gz'
>>> apply_xfm.run()
```

reference [a pathlike object or string representing an existing file] Reference image used for target space.

xfm_file [a pathlike object or string representing an existing file] Transform file.

in_files [a list of items which are a pathlike object or string representing an existing file] Files to be registered.

interp ['nearestneighbour' or 'spline'] Interpolation used.

out_files [a list of items which are a pathlike object or string representing a file] Transformed files.

BinaryThreshold

Link to code

Bases: nipype.interfaces.fsl.base.FSLCommand

Wrapped executable: `fslmaths`.

Use `fslmaths` to apply a threshold to an image in a variety of ways.

Examples

```
>>> from cmtklib.interfaces.fsl import BinaryThreshold
>>> thresh = BinaryThreshold()
>>> thresh.inputs.in_file = '/path/to/probseg.nii.gz'
>>> thresh.inputs.thresh = 0.5
>>> thresh.inputs.out_file = '/path/to/output_binseg.nii.gz'
>>> thresh.run()
```

in_file [a pathlike object or string representing an existing file] Image to operate on. Maps to a command-line argument: `%s` (position: 2).

out_file [a pathlike object or string representing a file] Image to write. Maps to a command-line argument: `%s` (position: 5).

thresh [a float] Threshold value. Maps to a command-line argument: `-thr %s` (position: 3).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

binarize [a boolean] Maps to a command-line argument: `-bin` (position: 4).

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

output_type ['NIFTI' or 'NIFTI_PAIR' or 'NIFTI_GZ' or 'NIFTI_PAIR_GZ'] FSL output type.

out_file [a pathlike object or string representing an existing file] Image written after calculations.

CreateAcqpFile

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Create an acquisition `Acqp` file for FSL `eddy`.

Note: This value can be extracted from dMRI data acquired on Siemens scanner

Examples

```
>>> from cmtklib.interfaces.fsl import CreateAcqpFile
>>> create_acqp = CreateAcqpFile()
>>> create_acqp.inputs.total_readout  = 0.28
>>> create_acqp.run()
```

total_readout : a float

acqp : a pathlike object or string representing an existing file

CreateIndexFile

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Create an index file for FSL `eddy` from a `mrtrix` diffusion gradient table.

Examples

```
>>> from cmtklib.interfaces.fsl import CreateIndexFile
>>> create_index = CreateIndexFile()
>>> create_index.inputs.in_grad_mrtrix  = 'grad.txt'
>>> create_index.run()
```

in_grad_mrtrix [a pathlike object or string representing an existing file] Input DWI gradient table in MRTrix format.

index : a pathlike object or string representing an existing file

Eddy

Link to code

Bases: nipype.interfaces.fsl.base.FSLCommand

Wrapped executable: `eddy`.

Performs eddy current distorsion correction using FSL `eddy`.

Example

```
>>> from cmtklib.interfaces import fsl
>>> eddyc = fsl.Eddy(in_file='diffusion.nii',
>>>                  bvecs='diffusion.bvecs',
>>>                  bvals='diffusion.bvals',
>>>                  out_file="diffusion_eddyc.nii")
>>> eddyc.run()
```

acqp [a pathlike object or string representing an existing file] File containing acquisition parameters. Maps to a command-line argument: `--acqp=%s` (position: 3).

bvals [a pathlike object or string representing an existing file] File containing the b-values for all volumes in –imain. Maps to a command-line argument: `--bvals=%s` (position: 5).

bvecs [a pathlike object or string representing an existing file] File containing the b-vectors for all volumes in –imain. Maps to a command-line argument: `--bvecs=%s` (position: 4).

in_file [a pathlike object or string representing an existing file] File containing all the images to estimate distortions for. Maps to a command-line argument: `--imain=%s` (position: 0).

index [a pathlike object or string representing an existing file] File containing indices for all volumes in –imain into –acqp and –topup. Maps to a command-line argument: `--index=%s` (position: 2).

mask [a pathlike object or string representing an existing file] Mask to indicate brain. Maps to a command-line argument: `--mask=%s` (position: 1).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

out_file [a pathlike object or string representing a file] Basename for output. Maps to a command-line argument: `--out=%s` (position: 6).

output_type ['NIFTI' or 'NIFTI_PAIR' or 'NIFTI_GZ' or 'NIFTI_PAIR_GZ'] FSL output type.

verbose [a boolean] Display debugging messages. Maps to a command-line argument: `--verbose` (position: 7).

bvecs_rotated [a pathlike object or string representing an existing file] Path/name of rotated DWI gradient bvecs file.

eddy_corrected [a pathlike object or string representing an existing file] Path/name of 4D eddy corrected DWI file.

EddyOpenMP

Link to code

Bases: nipype.interfaces.fsl.base.FSLCommand

Wrapped executable: `eddy_openmp`.

Performs eddy current distorsion correction using FSL `eddy_openmp`.

Example

```
>>> from cmtklib.interfaces import fsl
>>> eddyc = fsl.EddyOpenMP(in_file='diffusion.nii',
>>>                         bvecs='diffusion.bvecs',
>>>                         bvals='diffusion.bvals',
>>>                         out_file="diffusion_eddyc.nii")
>>> eddyc.run()
```

acqp [a pathlike object or string representing an existing file] File containing acquisition parameters. Maps to a command-line argument: `--acqp=%s` (position: 3).

bvals [a pathlike object or string representing an existing file] File containing the b-values for all volumes in –imain. Maps to a command-line argument: `--bvals=%s` (position: 5).

bvecs [a pathlike object or string representing an existing file] File containing the b-vectors for all volumes in –imain. Maps to a command-line argument: `--bvecs=%s` (position: 4).

in_file [a pathlike object or string representing an existing file] File containing all the images to estimate distortions for. Maps to a command-line argument: `--imain=%s` (position: 0).

index [a pathlike object or string representing an existing file] File containing indices for all volumes in –imain into –acqp and –topup. Maps to a command-line argument: `--index=%s` (position: 2).

mask [a pathlike object or string representing an existing file] Mask to indicate brain. Maps to a command-line argument: `--mask=%s` (position: 1).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

out_file [a pathlike object or string representing a file] Basename for output. Maps to a command-line argument: `--out=%s` (position: 6).

output_type ['NIFTI' or 'NIFTI_PAIR' or 'NIFTI_GZ' or 'NIFTI_PAIR_GZ'] FSL output type.

verbose [a boolean] Display debugging messages. Maps to a command-line argument: `--verbose` (position: 7).

bvecs_rotated [a pathlike object or string representing an existing file] Path/name of rotated DWI gradient bvecs file.

eddy_corrected [a pathlike object or string representing an existing file] Path/name of 4D eddy corrected DWI file.

FSLCreateHD

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `fslcreatehd`.

Calls the `fslcreatehd` command to create an image for space / dimension reference.

Examples

```
>>> from cmtklib.interfaces.fsl import FSLCreateHD
>>> fsl_create = FSLCreateHD()
>>> fsl_create.inputs.im_size = [256, 256, 256, 1]
>>> fsl_create.inputs.vox_size = [1, 1, 1]
>>> fsl_create.inputs.tr = 0
>>> fsl_create.inputs.origin = [0, 0, 0]
>>> fsl_create.inputs.datatype = '16' # 16: float
>>> fsl_create.inputs.out_filename = '/path/to/generated_image.nii.gz'
>>> fsl_create.run()
```

datatype [‘2’ or ‘4’ or ‘8’ or ‘16’ or ‘32’ or ‘64’] Datatype values: 2=char, 4=short, 8=int, 16=float, 64=double. Maps to a command-line argument: `%s` (position: 5).

im_size [a list of from 4 to 4 items which are an integer] Image size : xsize , ysize, zsize, tsize . Maps to a command-line argument: `%s` (position: 1).

origin [a list of from 3 to 3 items which are an integer] Origin coordinates : xorig, yorig, zorig. Maps to a command-line argument: `%s` (position: 4).

out_filename [a pathlike object or string representing a file]

the output temp reference image created.

Maps to a command-line argument: `%s` (position: 6).

tr [an integer] <tr>. Maps to a command-line argument: `%s` (position: 3).

vox_size [a list of from 3 to 3 items which are an integer] Voxel size : xvoxsize, yvoxsize, zvoxsize. Maps to a command-line argument: `%s` (position: 2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class ‘str’ and with values which are a bytes or None or a value of class ‘str’] Environment variables. (Nipype **default** value: `{}`)

out_file [a pathlike object or string representing an existing file] Path/name of the output reference image created.

MathsCommand

Link to code

Bases: nipype.interfaces.fsl.base.FSLCommand

Wrapped executable: `fslmaths`.

Calls the `fslmaths` command in a variety of ways.

Examples

```
>>> from cmtklib.interfaces.fsl import MathsCommand
>>> fsl_maths = MathsCommand()
>>> fsl_maths.inputs.in_file = '/path/to/image_with_nans.nii.gz'
>>> fsl_maths.inputs.nan2zeros = True
>>> fsl_maths.inputs.out_file = '/path/to/image_with_no_nans.nii.gz'
>>> fsl_maths.run()
```

in_file [a pathlike object or string representing an existing file] Image to operate on. Maps to a command-line argument: `%s` (position: 2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

internal_datatype ['float' or 'char' or 'int' or 'short' or 'double' or 'input'] Datatype to use for calculations (default is float). Maps to a command-line argument: `-dt %s` (position: 1).

nan2zeros [a boolean] Change NaNs to zeros before doing anything. Maps to a command-line argument: `-nan` (position: 3).

out_file [a pathlike object or string representing a file] Image to write. Maps to a command-line argument: `%s` (position: -2).

output_datatype ['float' or 'char' or 'int' or 'short' or 'double' or 'input'] Datatype to use for output (default uses input type). Maps to a command-line argument: `-odt %s` (position: -1).

output_type ['NIFTI' or 'NIFTI_PAIR' or 'NIFTI_GZ' or 'NIFTI_PAIR_GZ'] FSL output type.

out_file [a pathlike object or string representing an existing file] Image written after calculations.

Orient

Link to code

Bases: nipype.interfaces.fsl.base.FSLCommand

Wrapped executable: `fslorient`.

Use fslorient to get/set orientation information from an image's header.

Advanced tool that reports or sets the orientation information in a file. Note that only in NIfTI files can the orientation be changed - Analyze files are always treated as "radiological" (meaning that they could be simply rotated into the same alignment as the MNI152 standard images - equivalent to the appropriate sform or qform in a NIfTI file having a negative determinant).

Examples

```
>>> from cmtklib.interfaces.fsl import Orient
>>> fsl_orient = Orient()
>>> fsl_orient.inputs.in_file = 'input_image.nii.gz'
>>> fsl_orient.inputs.force_radiological = True
>>> fsl_orient.inputs.out_file = 'output_image.nii.gz'
>>> fsl_orient.run()
```

in_file [a pathlike object or string representing an existing file] Input image. Maps to a command-line argument: `%s` (position: 2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

copy_qform2sform [a boolean] Sets the sform equal to the qform - code and matrix. Maps to a command-line argument: `-copyqform2sform` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

copy_sform2qform [a boolean] Sets the qform equal to the sform - code and matrix. Maps to a command-line argument: `-copysform2qform` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

delete_orient [a boolean] Removes orient info from header. Maps to a command-line argument: `-deleteorient` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

force_neurological [a boolean] Makes FSL neurological header - not Analyze. Maps to a command-line argument: `-forceneurological` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

force_radiological [a boolean] Makes FSL radiological header. Maps to a command-line argument: `-forceradiological` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

get_orient [a boolean] Gets FSL left-right orientation. Maps to a command-line argument: `-getorient` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

get_qform [a boolean] Gets the 16 elements of the qform matrix. Maps to a command-line argument: `-getqform` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

get_qformcode [a boolean] Gets the qform integer code. Maps to a command-line argument: `-getqformcode` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

get_sform [a boolean] Gets the 16 elements of the sform matrix. Maps to a command-line argument: `-getsform` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

get_sformcode [a boolean] Gets the sform integer code. Maps to a command-line argument: `-getsformcode` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

output_type ['NIFTI' or 'NIFTI_PAIR' or 'NIFTI_GZ' or 'NIFTI_PAIR_GZ'] FSL output type.

set_qform [a list of from 16 to 16 items which are a float] <m11 m12 … m44> sets the 16 elements of the qform matrix. Maps to a command-line argument: `-setqform %f` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

set_qformcode [an integer] <code> sets qform integer code. Maps to a command-line argument: `-setqormcode %d` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

set_sform [a list of from 16 to 16 items which are a float] <m11 m12 … m44> sets the 16 elements of the sform matrix. Maps to a command-line argument: `-setsform %f` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

set_sformcode [an integer] <code> sets sform integer code. Maps to a command-line argument: `-setformcode %d` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

swap_orient [a boolean] Swaps FSL radiological and FSL neurological. Maps to a command-line argument: `-swaporient` (position: 1). Mutually **exclusive** with inputs: `get_orient`, `get_sform`, `get_qform`, `set_sform`, `set_qform`, `get_sformcode`, `get_qformcode`, `set_sformcode`, `set_qformcode`, `copy_sform2qform`, `copy_qform2sform`, `delete_orient`, `force_radiological`, `force_neurological`, `swap_orient`.

orient [a string] FSL left-right orientation.

out_file [a pathlike object or string representing an existing file] Image with modified orientation.

qform [a list of from 16 to 16 items which are a float] The 16 elements of the qform matrix.

qformcode [an integer] Qform integer code.

sform [a list of from 16 to 16 items which are a float] The 16 elements of the sform matrix.

sformcode [an integer] Sform integer code.

`Orient.aggregate_outputs(runtime=None, needed_outputs=None)`
Collate expected outputs and apply output traits validation.

cmtklib.interfaces.misc module

ConcatOutputsAsTuple

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Concatenate 2 different output file as a Tuple of 2 files.

Examples

```
>>> from cmtklib.interfaces.misc import ConcatOutputsAsTuple
>>> concat_outputs = ConcatOutputsAsTuple()
>>> concat_outputs.inputs.input1  = 'output_interface1.nii.gz'
>>> concat_outputs.inputs.input2  = 'output_interface2.nii.gz'
>>> concat_outputs.run()
```

input1 : a pathlike object or string representing an existing file input2 : a pathlike object or string representing an existing file

out_tuple : a tuple of the form: (a pathlike object or string representing an existing file, a pathlike object or string representing an existing file)

ExtractHeaderVoxel2WorldMatrix

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Write in a text file the voxel-to-world transform matrix from the heaer of a Nifti image.

Examples

```
>>> from cmtklib.interfaces.misc import ExtractHeaderVoxel2WorldMatrix
>>> extract_mat = ExtractHeaderVoxel2WorldMatrix()
>>> extract_mat.inputs.in_file = 'sub-01_T1w.nii.gz'
>>> extract_mat.run()
```

in_file [a pathlike object or string representing an existing file] Input image file.

out_matrix [a pathlike object or string representing an existing file] Output voxel to world affine transform file.

ExtractImageVoxelSizes

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Returns a list of voxel sizes from an image.

Examples

```
>>> from cmtklib.interfaces.misc import ExtractImageVoxelSizes
>>> extract_voxel_sizes = ExtractImageVoxelSizes()
>>> extract_voxel_sizes.inputs.in_file = 'sub-01_T1w.nii.gz'
>>> extract_voxel_sizes.run()
```

in_file : a pathlike object or string representing an existing file

voxel_sizes : a list of items which are any value

Rename001

Link to code

Bases: nipype.interfaces.utility.base.Rename

Change the name of a file based on a mapped format string.

To use additional inputs that will be defined at run-time, the class constructor must be called with the format template, and the fields identified will become inputs to the interface. Additionally, you may set the parse_string input, which will be run over the input filename with a regular expressions search, and will fill in additional input fields from matched groups. Fields set with inputs have precedence over fields filled in with the regexp match.

It corresponds to the `nipype.interfaces.utility.base.Rename` interface that has been modified to force hard link during copy

Examples

```
>>> from nipype.interfaces.utility import Rename
>>> rename1 = Rename()
>>> rename1.inputs.in_file = os.path.join(datadir, "zstat1.nii.gz") # datadir␣
↪is a directory with exemplary files, defined in conftest.py
>>> rename1.inputs.format_string = "Faces-Scenes.nii.gz"
>>> res = rename1.run()
>>> res.outputs.out_file
'Faces-Scenes.nii.gz"
```

```
>>> rename2 = Rename(format_string="%(subject_id)s_func_run%(run)02d")
>>> rename2.inputs.in_file = os.path.join(datadir, "functional.nii")
>>> rename2.inputs.keep_ext = True
>>> rename2.inputs.subject_id = "subj_201"
>>> rename2.inputs.run = 2
>>> res = rename2.run()
```

(continues on next page)

(continued from previous page)

```
>>> res.outputs.out_file
'subj_201_func_run02.nii'
```

```
>>> rename3 = Rename(format_string="%(subject_id)s_%(seq)s_run%(run)02d.nii")
>>> rename3.inputs.in_file = os.path.join(datadir, "func_epi_1_1.nii")
>>> rename3.inputs.parse_string = r"func_(?P<seq>\w*)_.*"
>>> rename3.inputs.subject_id = "subj_201"
>>> rename3.inputs.run = 2
>>> res = rename3.run()
>>> res.outputs.out_file
'subj_201_epi_run02.nii'
```

References

Adapted from https://github.com/nipy/nipype/blob/cd4c34d935a43812d1756482fdc4034844e485b8/nipype/interfaces/utility/base.py#L232-L272

format_string [a string] Python formatting string for output template.

in_file [a pathlike object or string representing an existing file] File to rename.

keep_ext [a boolean] Keep in_file extension, replace non-extension component of name.

parse_string [a string] Python regexp parse string to define replacement inputs.

use_fullpath [a boolean] Use full path as input to regex parser. (Nipype **default** value: `False`)

out_file [a pathlike object or string representing an existing file] Softlink to original file with new name.

cmtklib.interfaces.mne module

The MNE module provides Nipype interfaces for MNE tools missing in Nipype or modified.

CreateBEM

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Use MNE to create the BEM surfaces.

Examples

```
>>> from cmtklib.interfaces.mne import CreateBEM
>>> create_bem = CreateBEM()
>>> create_bem.inputs.fs_subject = 'sub-01'
>>> create_bem.inputs.fs_subjects_dir = '/path/to/bids_dataset/derivatives/
↪freesurfer-7.1.1'
>>> create_bem.inputs.out_bem_fname = 'bem.fif'
>>> create_bem.run()
```

References

- https://mne.tools/stable/generated/mne.bem.make_watershed_bem.html
- https://mne.tools/stable/generated/mne.make_bem_model.html
- https://mne.tools/stable/generated/mne.write_bem_solution.html

fs_subject [a string] FreeSurfer subject ID.

fs_subjects_dir [a string or os.PathLike object referring to an existing directory] Freesurfer subjects (derivatives) directory.

out_bem_fname [a string] Name of output BEM file in fif format.

bem_file [a string or os.PathLike object] Path to output BEM file in fif format.

CreateCov

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Use MNE to create the noise covariance matrix.

Examples

```
>>> from cmtklib.interfaces.mne import CreateCov
>>> create_cov = CreateCov()
>>> create_cov.inputs.epochs_file = '/path/to/sub-01_epo.fif'
>>> create_cov.inputs.out_noise_cov_fname = 'sub-01_noisecov.fif'
>>> create_cov.run()
```

References

- https://mne.tools/stable/generated/mne.Covariance.html

epochs_file [a string or os.PathLike object referring to an existing file] Eeg * epochs in .set format.

out_noise_cov_fname [a string] Name of output file to save noise covariance matrix in fif format.

noise_cov_file [a string or os.PathLike object] Location and name to store noise covariance matrix in fif format.

CreateFwd

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Use MNE to calculate the forward solution.

Examples

```
>>> from cmtklib.interfaces.mne import CreateFwd
>>> create_fwd = CreateFwd()
>>> create_fwd.inputs.epochs_file = '/path/to/sub-01_epo.fif'
>>> create_fwd.inputs.out_fwd_fname = 'sub-01_fwd.fif'
>>> create_fwd.inputs.src_file = '/path/to/sub-01_src.fif'
>>> create_fwd.inputs.bem_file = '/path/to/sub-01_bem.fif'
>>> create_fwd.inputs.trans_file = '/path/to/sub-01_trans.fif'
>>> create_fwd.run()
```

References

- https://mne.tools/stable/generated/mne.make_forward_solution.html

bem_file [a string or os.PathLike object referring to an existing file] Boundary surfaces for MNE head model in fif format.

epochs_file [a string or os.PathLike object referring to an existing file] Eeg * epochs in .fif format, containing information about electrode montage.

src_file [a string or os.PathLike object referring to an existing file] Source space file in fif format.

out_fwd_fname [a string] Name of output forward solution file created with MNE.

trans_file [a string or os.PathLike object referring to an existing file] Trans.fif file containing co-registration information (electrodes x MRI).

fwd_file [a string or os.PathLike object] Path to generated forward solution file in fif format.

CreateSrc

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Use MNE to set up bilateral hemisphere surface-based source space with subsampling and write source spaces to a file.

Examples

```
>>> from cmtklib.interfaces.mne import CreateSrc
>>> create_src = CreateSrc()
>>> create_src.inputs.fs_subject = 'sub-01'
>>> create_src.inputs.fs_subjects_dir = '/path/to/bids_dataset/derivatives/
↪freesurfer-7.1.1'
>>> create_src.inputs.out_src_fname = 'sub-01_src.fif'
>>> create_src.run()
```

References

- https://mne.tools/stable/generated/mne.setup_source_space.html
- https://mne.tools/stable/generated/mne.write_source_spaces.html

fs_subject [a string] FreeSurfer subject ID.

fs_subjects_dir [a string or os.PathLike object referring to an existing directory] Freesurfer subjects (derivatives) directory.

out_src_fname [a string] Name of output source space file created with MNE.

overwrite [a boolean] Overwrite source space file if already existing.

src_file [a string or os.PathLike object] Path to output source space files in fif format.

EEGLAB2fif

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Use MNE to convert EEG data from EEGlab to MNE format.

Examples

```
>>> from cmtklib.interfaces.mne import EEGLAB2fif
>>> eeglab2fif = EEGLAB2fif()
>>> eeglab2fif.inputs.eeg_ts_file = ['sub-01_task-faces_desc-preproc_eeg.set']
>>> eeglab2fif.inputs.events_file = ['sub-01_task-faces_events.tsv']
>>> eeglab2fif.inputs.out_epochs_fif_fname = 'sub-01_epo.fif'
>>> eeglab2fif.inputs.electrodes_file = 'sub-01_eeg.xyz'
>>> eeglab2fif.inputs.event_ids = {"SCRAMBLED":0, "FACES":1}
>>> eeglab2fif.inputs.t_min = -0.2
>>> eeglab2fif.inputs.t_max = 0.6
>>> eeglab2fif.run()
```

References

- https://mne.tools/stable/generated/mne.read_epochs_eeglab.html
- https://mne.tools/stable/generated/mne.channels.make_dig_montage.html
- https://mne.tools/stable/generated/mne.Epochs.html?highlight=set_montage#mne.Epochs.set_montage
- https://mne.tools/stable/generated/mne.Epochs.html?highlight=set_montage#mne.Epochs.save

eeg_ts_file [a string or os.PathLike object referring to an existing file] Eeg * epochs in .set format.

events_file [a string or os.PathLike object referring to an existing file] Epochs metadata in _behav.txt.

out_epochs_fif_fname [a string] Output filename for eeg * epochs in .fif format, e.g. sub-01_epo.fif.

electrodes_file [a string or os.PathLike object referring to an existing file] Positions of EEG electrodes in a txt file.

event_ids [a dictionary with keys which are any value and with values which are any value] The id of the events to consider in `dict` form. The keys of the `dict` can later be used to access associated events. If None, all events will be used and a dict is created with string integer names corresponding to the event id integers.

t_max [a float] End time of the epochs in seconds, relative to the time-locked event.

t_min [a float] Start time of the epochs in seconds, relative to the time-locked event.

epochs_file [a string or os.PathLike object referring to an existing file] Eeg * epochs in .fif format.

MNEInverseSolutionROI

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Use MNE to convert EEG data from EEGlab to MNE format.

Examples

```
>>> from cmtklib.interfaces.mne import MNEInverseSolutionROI
>>> inv_sol = MNEInverseSolutionROI()
>>> inv_sol.inputs.esi_method_snr = 3.0
>>> inv_sol.inputs.fs_subject = 'sub-01'
>>> inv_sol.inputs.fs_subjects_dir = '/path/to/bids_dataset/derivatives/
↪freesurfer-7.1.1'
>>> inv_sol.inputs.epochs_file = '/path/to/sub-01_epo.fif'
>>> inv_sol.inputs.src_file = '/path/to/sub-01_src.fif'
>>> inv_sol.inputs.bem_file = '/path/to/sub-01_bem.fif'
>>> inv_sol.inputs.noise_cov_file = '/path/to/sub-01_noisecov.fif'
>>> inv_sol.inputs.fwd_file = '/path/to/sub-01_fwd.fif'
>>> inv_sol.inputs.atlas_annot = 'lausanne2018.scale1'
>>> inv_sol.inputs.out_roi_ts_fname_prefix = 'sub-01_atlas-L2018_res-scale1_
↪desc-epo_timeseries'
>>> inv_sol.inputs.out_inv_fname = 'sub-01_inv.fif'
>>> inv_sol.run()
```

References

- https://mne.tools/stable/generated/mne.read_forward_solution.html
- https://mne.tools/stable/generated/mne.minimum_norm.make_inverse_operator.html
- https://mne.tools/stable/generated/mne.minimum_norm.apply_inverse_epochs.html
- https://mne.tools/stable/generated/mne.read_labels_from_annot.html
- https://mne.tools/stable/generated/mne.extract_label_time_course.html

bem_file [a string or os.PathLike object referring to an existing file] Surfaces for head model in fif format.

epochs_file [a string or os.PathLike object referring to an existing file] Eeg * epochs in .fif format.

fs_subject [a string] FreeSurfer subject ID.

fs_subjects_dir [a string or os.PathLike object referring to an existing directory] Freesurfer subjects (derivatives) directory.

fwd_file [a string or os.PathLike object] Forward solution in fif format.

noise_cov_file [a string or os.PathLike object referring to an existing file] Noise covariance matrix in fif format.

out_inv_fname [a string] Output filename for inverse operator in fif format.

src_file [a string or os.PathLike object referring to an existing file] Source space created with MNE in fif format.

atlas_annot ['aparc' or 'lausanne2018.scale1' or 'lausanne2018.scale2' or 'lausanne2018.scale3' or 'lausanne2018.scale4' or 'lausanne2018.scale5'] The parcellation to use, e.g., 'aparc', 'lausanne2018.scale1', 'lausanne2018.scale2', 'lausanne2018.scale3', 'lausanne2018.scale4' or'lausanne2018.scale5'.

esi_method ['sLORETA' or 'eLORETA' or 'MNE' or 'dSPM'] Use minimum norm 1, dSPM 2, sLORETA (default) 3, or eLORETA 4.

esi_method_snr [a float] SNR value such as the ESI method regularization weight lambda2 is set to `1.0 / esi_method_snr ** 2`.

out_roi_ts_fname_prefix [a string] Output filename prefix (no extension) for rois * time series in .npy and .mat formats.

inv_file [a string or os.PathLike object] Path to output inverse operator file in fif format.

roi_ts_mat_file [a string or os.PathLike object] Path to output ROI time series file in .mat format.

roi_ts_npy_file [a string or os.PathLike object] Path to output ROI time series file in .npy format.

MNESpectralConnectivity

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Use MNE to compute frequency- and time-frequency-domain connectivity measures.

Examples

```
>>> from cmtklib.interfaces.mne import MNESpectralConnectivity
>>> eeg_cmat = MNESpectralConnectivity()
>>> eeg_cmat.inputs.fs_subject = 'sub-01'
>>> eeg_cmat.inputs.fs_subjects_dir = '/path/to/bids_dataset/derivatives/
↪freesurfer-7.1.1'
>>> eeg_cmat.inputs.atlas_annot = 'lausanne2018.scale1'
>>> eeg_cmat.inputs.connectivity_metrics = ['imcoh', 'pli', 'wpli']
>>> eeg_cmat.inputs.output_types = ['tsv', 'gpickle', 'mat', 'graphml']
>>> eeg_cmat.inputs.epochs_file = '/path/to/sub-01_epo.fif'
>>> eeg_cmat.inputs.roi_ts_file = '/path/to/sub-01_timeseries.npy'
>>> eeg_cmat.run()
```

References

- https://mne.tools/mne-connectivity/stable/generated/mne_connectivity.spectral_connectivity_epochs.html

fs_subject [a string] FreeSurfer subject ID.

fs_subjects_dir [a string or os.PathLike object referring to an existing directory] Freesurfer subjects (derivatives) directory.

atlas_annot ['aparc' or 'lausanne2018.scale1' or 'lausanne2018.scale2' or 'lausanne2018.scale3' or 'lausanne2018.scale4' or 'lausanne2018.scale5'] The parcellation to use, e.g., 'aparc', 'lausanne2018.scale1', 'lausanne2018.scale2', 'lausanne2018.scale3', 'lausanne2018.scale4' or'lausanne2018.scale5'.

connectivity_metrics [a list of items which are any value] Set of frequency- and time-frequency-domain connectivity metrics to compute.

epochs_file [a pathlike object or string representing an existing file] Epochs file in fif format.

out_cmat_fname [a string] Basename of output connectome file (without any extension).

output_types [a list of items which are any value] Set of format to save output connectome files.

roi_ts_file [a pathlike object or string representing an existing file] Extracted ROI time courses from ESI in .npy format.

roi_volume_tsv_file [a pathlike object or string representing an existing file] Index / label atlas mapping file in .tsv format accordingly to BIDS.

connectivity_matrices [a list of items which are a pathlike object or string representing a file] Connectivity matrices.

cmtklib.interfaces.mrtrix3 module

The MRTrix3 module provides Nipype interfaces for MRTrix3 tools missing in Nipype or modified.

ApplymultipleMRConvert

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Apply `mrconvert` tool to multiple images.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> mrconvert = mrt.ApplymultipleMRConvert()
>>> mrconvert.inputs.in_files = ['dwi_FA.mif','dwi_MD.mif']
>>> mrconvert.inputs.extension = 'nii'
>>> mrconvert.run()
```

extension ['mif' or 'nii' or 'float' or 'char' or 'short' or 'int' or 'long' or 'double'] "i.e. Bfloat". Can be "char", "short", "int", "long", "float" or "double". (Nipype **default** value: `mif`)

in_files [a list of items which are a pathlike object or string representing an existing file] Files to be registered.

output_datatype ['float32' or 'float32le' or 'float32be' or 'float64' or 'float64le' or 'float64be' or 'int64' or 'uint64' or 'int64le' or 'uint64le' or 'int64be' or 'uint64be' or 'int32' or 'uint32' or 'int32le' or 'uint32le' or 'int32be' or 'uint32be' or 'int16' or 'uint16' or 'int16le' or 'uint16le' or 'int16be' or 'uint16be' or 'cfloat32' or 'cfloat32le' or 'cfloat32be' or 'cfloat64' or 'cfloat64le' or 'cfloat64be' or 'int8' or 'uint8' or 'bit'] Specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be, uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit. Maps to a command-line argument: `-datatype %s` (position: 2).

stride [a list of from 3 to 4 items which are an integer] Three to four comma-separated numbers specifying the strides of the output data in memory. The actual strides produced will depend on whether the output image format can support it.. Maps to a command-line argument: `-stride %s` (position: 3).

converted_files [a list of items which are a pathlike object or string representing a file] Output files.

ApplymultipleMRCrop

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Apply MRCrop to a list of images.

Example

```
>>> from cmtklib.interfaces.mrtrix3 import ApplymultipleMRCrop
>>> multi_crop = ApplymultipleMRCrop()
>>> multi_crop.inputs.in_files = ['/sub-01_atlas-L2018_desc-scale1_dseg.nii.gz
↪',
>>>                               'sub-01_atlas-L2018_desc-scale2_dseg.nii.gz',
>>>                               'sub-01_atlas-L2018_desc-scale3_dseg.nii.gz',
>>>                               'sub-01_atlas-L2018_desc-scale4_dseg.nii.gz',
>>>                               'sub-01_atlas-L2018_desc-scale5_dseg.nii.gz']
>>> multi_crop.inputs.template_image = 'sub-01_T1w.nii.gz'
>>> multi_crop.run()
```

See also:

`cmtklib.interfaces.mrtrix3.MRCrop`

template_image [a pathlike object or string representing an existing file] Template image.

in_files [a list of items which are a pathlike object or string representing an existing file] Files to be cropped.

out_files [a list of items which are a pathlike object or string representing a file] Cropped files.

ApplymultipleMRTransforms

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Apply MRTransform to a list of images.

Example

```
>>> from cmtklib.interfaces.mrtrix3 import ApplymultipleMRTransforms
>>> multi_transform = ApplymultipleMRTransforms()
>>> multi_transform.inputs.in_files = ['/sub-01_atlas-L2018_desc-scale1_dseg.
↪nii.gz',
>>>                                      'sub-01_atlas-L2018_desc-scale2_dseg.
↪nii.gz',
>>>                                      'sub-01_atlas-L2018_desc-scale3_dseg.
↪nii.gz',
>>>                                      'sub-01_atlas-L2018_desc-scale4_dseg.
↪nii.gz',
>>>                                      'sub-01_atlas-L2018_desc-scale5_dseg.
↪nii.gz']
>>> multi_transform.inputs.template_image = 'sub-01_T1w.nii.gz'
>>> multi_transform.run()
```

See also:

`cmtklib.interfaces.mrtrix3.MRTransform`

template_image [a pathlike object or string representing an existing file] Template image.

in_files [a list of items which are a pathlike object or string representing an existing file] Files to be transformed.

out_files [a list of items which are a pathlike object or string representing a file] Transformed files.

ConstrainedSphericalDeconvolution

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `dwi2fod`.

Perform non-negativity constrained spherical deconvolution using `dwi2fod`.

Note that this program makes use of implied symmetries in the diffusion profile. First, the fact the signal attenuation profile is real implies that it has conjugate symmetry, i.e. Y(l,-m) = Y(l,m)* (where * denotes the complex conjugate). Second, the diffusion profile should be antipodally symmetric (i.e. S(x) = S(-x)), implying that all odd l components should be zero. Therefore, this program only computes the even elements. Note that the spherical harmonics equations used here differ slightly from those conventionally used, in that the (-1)^m factor has been omitted. This should be taken into account in all subsequent calculations. Each volume in the output image corresponds to a different spherical harmonic component, according to the following convention:

- [0] Y(0,0)
- [1] Im {Y(2,2)}

- [2] Im {Y(2,1)}
- [3] Y(2,0)
- [4] Re {Y(2,1)}
- [5] Re {Y(2,2)}
- [6] Im {Y(4,4)}
- [7] Im {Y(4,3)}

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> csdeconv = mrt.ConstrainedSphericalDeconvolution()
>>> csdeconv.inputs.in_file = 'dwi.mif'
>>> csdeconv.inputs.encoding_file = 'encoding.txt'
>>> csdeconv.run()
```

algorithm [‘csd’] Use CSD algorithm for FOD estimation. Maps to a command-line argument: `%s` (position: -4).

in_file [a pathlike object or string representing an existing file] Diffusion-weighted image. Maps to a command-line argument: `%s` (position: -3).

response_file [a pathlike object or string representing an existing file] The diffusion-weighted signal response function for a single fibre population (see EstimateResponse). Maps to a command-line argument: `%s` (position: -2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

directions_file [a pathlike object or string representing an existing file] A text file containing the [el az] pairs for the directions: Specify the directions over which to apply the non-negativity constraint (by default, the built-in 300 direction set is used). Maps to a command-line argument: `-directions %s` (position: -2).

encoding_file [a pathlike object or string representing an existing file] Gradient encoding, supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units (1000 s/mm^2). See FSL2MRTrix. Maps to a command-line argument: `-grad %s` (position: 1).

environ [a dictionary with keys which are a bytes or None or a value of class ‘str’ and with values which are a bytes or None or a value of class ‘str’] Environment variables. (Nipype **default** value: `{}`)

filter_file [a pathlike object or string representing an existing file] A text file containing the filtering coefficients for each even harmonic order.the linear frequency filtering parameters used for the initial linear spherical deconvolution step (default = [1 1 1 0 0]). Maps to a command-line argument: `-filter %s` (position: -2).

iterations [an integer] The maximum number of iterations to perform for each voxel (default = 50). Maps to a command-line argument: `-niter %s`.

lambda_value [a float] The regularisation parameter lambda that controls the strength of the constraint (default = 1.0). Maps to a command-line argument: `-norm_lambda %s`.

mask_image [a pathlike object or string representing an existing file] Only perform computation within the specified binary brain mask image. Maps to a command-line argument: `-mask %s` (position: 2).

maximum_harmonic_order [an integer] Set the maximum harmonic order for the output series. By default, the program will use the highest possible lmax given the number of diffusion-weighted images. Maps to a command-line argument: `-lmax %s`.

out_filename [a pathlike object or string representing a file] Output filename. Maps to a command-line argument: `%s` (position: -1).

threshold_value [a float] The threshold below which the amplitude of the FOD is assumed to be zero, expressed as a fraction of the mean value of the initial FOD (default = 0.1). Maps to a command-line argument: `-threshold %s`.

spherical_harmonics_image [a pathlike object or string representing an existing file] Spherical harmonics image.

DWI2Tensor

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `dwi2tensor`.

Converts diffusion-weighted images to tensor images using `dwi2tensor`.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> dwi2tensor = mrt.DWI2Tensor()
>>> dwi2tensor.inputs.in_file = 'dwi.mif'
>>> dwi2tensor.inputs.encoding_file = 'encoding.txt'
>>> dwi2tensor.run()
```

in_file [a list of items which are any value] Diffusion-weighted images. Maps to a command-line argument: `%s` (position: -2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

debug [a boolean] Display debugging messages. Maps to a command-line argument: `-debug` (position: 1).

encoding_file [a pathlike object or string representing a file] Encoding file, , supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units (1000 s/mm^2). See FSL2MRTrix(). Maps to a command-line argument: `-grad %s` (position: 2).

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

ignore_slice_by_volume [a list of from 2 to 2 items which are an integer] Requires two values (i.e. [34 1] for [Slice Volume] Ignores the image slices specified when computing the tensor. Slice here means the z coordinate of the slice to be ignored. Maps to a command-line argument: `-ignoreslices %s` (position: 2).

ignore_volumes [a list of at least 1 items which are an integer] Requires two values (i.e. [2 5 6] for [Volumes] Ignores the image volumes specified when computing the tensor. Maps to a command-line argument: `-ignorevolumes %s` (position: 2).

in_mask_file [a pathlike object or string representing an existing file] Input DWI mask. Maps to a command-line argument: `-mask %s` (position: -3).

out_filename [a pathlike object or string representing a file] Output tensor filename. Maps to a command-line argument: `%s` (position: -1).

quiet [a boolean] Do not display information messages or progress status. Maps to a command-line argument: `-quiet` (position: 1).

tensor [a pathlike object or string representing an existing file] Path/name of output diffusion tensor image.

DWIBiasCorrect

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `dwibiascorrect`.

Correct for bias field in diffusion MRI data using the `dwibiascorrect` tool.

Example

```
>>> from cmtklib.interfaces.mrtrix3 import DWIBiasCorrect
>>> dwi_biascorr = DWIBiasCorrect()
>>> dwi_biascorr.inputs.in_file = 'sub-01_dwi.nii.gz'
>>> dwi_biascorr.inputs.use_ants = True
>>> dwi_biascorr.run()
```

in_file [a pathlike object or string representing an existing file] The input image series to be corrected. Maps to a command-line argument: `%s` (position: -2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

debug [a boolean] Display debugging messages. Maps to a command-line argument: `-debug` (position: 5).

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

force_writing [a boolean] Force file overwriting. Maps to a command-line argument: `-force` (position: 4).

mask [a pathlike object or string representing a file] Manually provide a mask image for bias field estimation (optional). Maps to a command-line argument: `-mask %s` (position: 2).

out_bias [a pathlike object or string representing a file] Output the estimated bias field. Maps to a command-line argument: `-bias %s` (position: 3).

out_file [a pathlike object or string representing a file] The output corrected image series. Maps to a command-line argument: `%s` (position: -1).

use_ants [a boolean] Use ANTS N4 to estimate the inhomogeneity field. Maps to a command-line argument: `ants` (position: 1). Mutually **exclusive** with inputs: `use_ants`, `use_fsl`.

use_fsl [a boolean] Use FSL FAST to estimate the inhomogeneity field. Maps to a command-line argument: `fsl` (position: 1). Mutually **exclusive** with inputs: `use_ants`, `use_fsl`.

out_bias [a pathlike object or string representing an existing file] Output estimated bias field.

out_file [a pathlike object or string representing an existing file] Output corrected DWI image.

DWIDenoise

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `dwidenoise`.

Denoise diffusion MRI data using the `dwidenoise` tool.

Example

```
>>> from cmtklib.interfaces.mrtrix3 import DWIDenoise
>>> dwi_denoise = DWIDenoise()
>>> dwi_denoise.inputs.in_file = 'sub-01_dwi.nii.gz'
>>> dwi_denoise.inputs.out_file = 'sub-01_desc-denoised_dwi.nii.gz'
>>> dwi_denoise.inputs.out_noisemap = 'sub-01_mod-dwi_noisemap.nii.gz'
>>> dwi_denoise.run()
```

in_file [a pathlike object or string representing an existing file] Input diffusion-weighted image filename. Maps to a command-line argument: `%s` (position: -2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

debug [a boolean] Display debugging messages. Maps to a command-line argument: `-debug` (position: 5).

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

extent_window [a list of from 3 to 3 items which are a float] Three comma-separated numbers giving the window size of the denoising filter. Maps to a command-line argument: `-extent %s` (position: 2).

force_writing [a boolean] Force file overwriting. Maps to a command-line argument: `-force` (position: 4).

mask [a pathlike object or string representing a file] Only perform computation within the specified binary brain mask image. (optional). Maps to a command-line argument: `-mask %s` (position: 1).

out_file [a pathlike object or string representing a file] Output denoised DWI image filename. Maps to a command-line argument: `%s` (position: -1).

out_noisemap [a pathlike object or string representing a file] Output noise map filename. Maps to a command-line argument: `-noise %s` (position: 3).

out_file [a pathlike object or string representing an existing file] Output denoised DWI image.

out_noisemap [a pathlike object or string representing an existing file] Output noise map (if generated).

Erode

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `maskfilter`.

Erode (or dilates) a mask (i.e. binary) image using the `maskfilter` tool.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> erode = mrt.Erode()
>>> erode.inputs.in_file = 'mask.mif'
>>> erode.run()
```

in_file [a pathlike object or string representing an existing file] Input mask image to be eroded. Maps to a command-line argument: `%s` (position: -3).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

debug [a boolean] Display debugging messages. Maps to a command-line argument: `-debug` (position: 1).

dilate [a boolean] Perform dilation rather than erosion. Maps to a command-line argument: `-dilate` (position: 1).

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

filtertype ['clean' or 'connect' or 'dilate' or 'erode' or 'median'] The type of filter to be applied (clean, connect, dilate, erode, median). Maps to a command-line argument: `%s` (position: -2).

number_of_passes [an integer] The number of passes (default: 1). Maps to a command-line argument: `-npass %s`.

out_filename [a pathlike object or string representing a file] Output image filename. Maps to a command-line argument: `%s` (position: -1).

quiet [a boolean] Do not display information messages or progress status. Maps to a command-line argument: `-quiet` (position: 1).

out_file [a pathlike object or string representing an existing file] The output image.

EstimateResponseForSH

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `dwi2response`.

Estimates the fibre response function for use in spherical deconvolution using `dwi2response`.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> estresp = mrt.EstimateResponseForSH()
>>> estresp.inputs.in_file = 'dwi.mif'
>>> estresp.inputs.mask_image = 'dwi_WMProb.mif'
>>> estresp.inputs.encoding_file = 'encoding.txt'
>>> estresp.run()
```

encoding_file [a pathlike object or string representing an existing file] Gradient encoding, supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units (1000 s/mm^2). See FSL2MRTrix. Maps to a command-line argument: `-grad %s` (position: -2).

in_file [a pathlike object or string representing an existing file] Diffusion-weighted images. Maps to a command-line argument: `%s` (position: 2).

mask_image [a pathlike object or string representing an existing file] Only perform computation within the specified binary brain mask image. Maps to a command-line argument: `-mask %s` (position: -1).

algorithm ['dhollander' or 'fa' or 'manual' or 'msmt_5tt' or 'tax' or 'tournier'] Select the algorithm to be used to derive the response function; additional details and options become available once an algorithm is nominated. Options are: dhollander, fa, manual, msmt_5tt, tax, tournier. Maps to a command-line argument: `%s` (position: 1).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

debug [a boolean] Display debugging messages. Maps to a command-line argument: `-debug`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

maximum_harmonic_order [an integer] Set the maximum harmonic order for the output series. By default, the program will use the highest possible lmax given the number of diffusion-weighted images. Maps to a command-line argument: `-lmax %s` (position: -3).

out_filename [a pathlike object or string representing a file] Output filename. Maps to a command-line argument: `%s` (position: 3).

quiet [a boolean] Do not display information messages or progress status. Maps to a command-line argument: `-quiet`.

response [a pathlike object or string representing an existing file] Spherical harmonics image.

ExtractFSLGrad

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `mrinfo`.

Use `mrinfo` to extract FSL gradient.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> fsl_grad = mrt.ExtractFSLGrad()
>>> fsl_grad.inputs.in_file = 'sub-01_dwi.mif'
>>> fsl_grad.inputs.out_grad_fsl = ['sub-01_dwi.bvecs', 'sub-01_dwi.bvals']
>>> fsl_grad.run()
```

in_file [a pathlike object or string representing an existing file] Input images to be read. Maps to a command-line argument: `%s` (position: -2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

out_grad_fsl [a tuple of the form: (a pathlike object or string representing a file, a pathlike object or string representing a file)] Export the DWI gradient table to files in FSL (bvecs / bvals) format. Maps to a command-line argument: `-export_grad_fsl %s %s`.

out_grad_fsl [a tuple of the form: (a pathlike object or string representing an existing file, a pathlike object or string representing an existing file)] Outputs [bvecs, bvals] DW gradient scheme (FSL format) if set.

ExtractMRTrixGrad

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `mrinfo`.

Use `mrinfo` to extract mrtrix gradient text file.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> mrtrix_grad = mrt.ExtractMRTrixGrad()
>>> mrtrix_grad.inputs.in_file = 'sub-01_dwi.mif'
>>> mrtrix_grad.inputs.out_grad_mrtrix = 'sub-01_gradient.txt'
>>> mrtrix_grad.run()
```

in_file [a pathlike object or string representing an existing file] Input images to be read. Maps to a command-line argument: `%s` (position: -2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

out_grad_mrtrix [a pathlike object or string representing a file] Export the DWI gradient table to file in MRtrix format. Maps to a command-line argument: `-export_grad_mrtrix %s`.

out_grad_mrtrix [a pathlike object or string representing a file] Output MRtrix gradient text file if set.

FilterTractogram

Link to code

Bases: *MRTrix3Base*

Wrapped executable: `tcksift`.

Spherical-deconvolution informed filtering of tractograms using `tcksift` [Smith2013SIFT].

References

Example

```
>>> import cmtklib.interfaces.mrtrix3 as cmp_mrt
>>> mrtrix_sift = cmp_mrt.FilterTractogram()
>>> mrtrix_sift.inputs.in_tracks = 'tractogram.tck'
>>> mrtrix_sift.inputs.in_fod = 'spherical_harmonics_image.nii.gz'
>>> mrtrix_sift.inputs.out_file = 'sift_tractogram.tck'
>>> mrtrix_sift.run()
```

in_fod [a pathlike object or string representing an existing file] Input image containing the spherical harmonics of the fibre orientation distributions. Maps to a command-line argument: `%s` (position: -2).

in_tracks [a pathlike object or string representing an existing file] Input track file in TCK format. Maps to a command-line argument: `%s` (position: -3).

act_file [a pathlike object or string representing an existing file] ACT 5TT image file. Maps to a command-line argument: `-act %s` (position: -4).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

out_file [a pathlike object or string representing a file] Output filtered tractogram. Maps to a command-line argument: `%s` (position: -1).

out_tracks [a pathlike object or string representing an existing file] Output filtered tractogram.

Generate5tt

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `5ttgen`.

Generate a 5TT image suitable for ACT using the selected algorithm using `5ttgen`.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> gen5tt = mrt.Generate5tt()
>>> gen5tt.inputs.in_file = 'T1.nii.gz'
>>> gen5tt.inputs.algorithm = 'fsl'
>>> gen5tt.inputs.out_file = '5tt.mif'
>>> gen5tt.cmdline
'5ttgen fsl T1.nii.gz 5tt.mif'
>>> gen5tt.run()
```

algorithm [‘fsl’ or ‘gif’ or ‘freesurfer’ or ‘hsvs’] Tissue segmentation algorithm. Maps to a command-line argument: `%s` (position: -3).

in_file [a pathlike object or string representing an existing file] Input image. Maps to a command-line argument: `-nocrop -sgm_amyg_hipp %s` (position: -2).

out_file [a pathlike object or string representing a file] Output image. Maps to a command-line argument: `%s` (position: -1).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class ‘str’ and with values which are a bytes or None or a value of class ‘str’] Environment variables. (Nipype **default** value: `{}`)

out_file [a pathlike object or string representing an existing file] Output image.

GenerateGMWMInterface

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `5tt2gmwmi`.

Generate a grey matter-white matter interface mask from the 5TT image using `5tt2gmwmi`.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as cmp_mrt
>>> genWMGMI = cmp_mrt.Generate5tt()
>>> genWMGMI.inputs.in_file = '5tt.mif'
>>> genWMGMI.inputs.out_file = 'gmwmi.mif'
>>> genGMWMI.run()
```

in_file [a pathlike object or string representing an existing file] Input 5TT image. Maps to a command-line argument: `%s` (position: -2).

out_file [a pathlike object or string representing a file] Output GW/WM interface image. Maps to a command-line argument: `%s` (position: -1).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class ‘str’ and with values which are a bytes or None or a value of class ‘str’] Environment variables. (Nipype **default** value: `{}`)

out_file [a pathlike object or string representing an existing file] Output image.

MRConvert

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `mrconvert`.

Perform conversion with `mrconvert` between different file types and optionally extract a subset of the input image.

If used correctly, this program can be a very useful workhorse. In addition to converting images between different formats, it can be used to extract specific studies from a data set, extract a specific region of interest, flip the images, or to scale the intensity of the images.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> mrconvert = mrt.MRConvert()
>>> mrconvert.inputs.in_file = 'dwi_FA.mif'
>>> mrconvert.inputs.out_filename = 'dwi_FA.nii'
>>> mrconvert.run()
```

in_dir [a pathlike object or string representing an existing directory] Directory containing DICOM files. Maps to a command-line argument: `%s` (position: -2). Mutually **exclusive** with inputs: `in_file`, `in_dir`.

in_file [a pathlike object or string representing an existing file] Voxel-order data filename. Maps to a command-line argument: `%s` (position: -2). Mutually **exclusive** with inputs: `in_file`, `in_dir`.

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

extension ['mif' or 'nii' or 'float' or 'char' or 'short' or 'int' or 'long' or 'double'] "i.e. Bfloat". Can be "char", "short", "int", "long", "float" or "double". (Nipype **default** value: `mif`)

extract_at_axis [1 or 2 or 3] Extract data only at the coordinates specified.This option specifies the Axis. Must be used in conjunction with extract_at_coordinate. . Maps to a command-line argument: `-coord %s` (position: 1).

extract_at_coordinate [a list of from 1 to 3 items which are an integer] Extract data only at the coordinates specified. This option specifies the coordinates. Must be used in conjunction with extract_at_axis. Three comma-separated numbers giving the size of each voxel in mm. Maps to a command-line argument: `%s` (position: 2).

force_writing [a boolean] Force file overwriting. Maps to a command-line argument: `-force`.

grad [a pathlike object or string representing an existing file] Gradient encoding, supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units (1000 s/mm^2). See FSL2MRTrix. Maps to a command-line argument: `-grad %s` (position: 9).

grad_fsl [a tuple of the form: (a pathlike object or string representing an existing file, a pathlike object or string representing an existing file)] [bvecs, bvals] DW gradient scheme (FSL format). Maps to a command-line argument: `-fslgrad %s %s`.

layout ['nii' or 'float' or 'char' or 'short' or 'int' or 'long' or 'double'] Specify the layout of the data in memory. The actual layout produced will depend on whether the output image format can support it. Maps to a command-line argument: `-output %s` (position: 5).

offset_bias [a float] Apply offset to the intensity values. Maps to a command-line argument: `-scale %d` (position: 7).

out_filename [a pathlike object or string representing a file] Output filename. Maps to a command-line argument: `%s` (position: -1).

output_datatype ['float32' or 'float32le' or 'float32be' or 'float64' or 'float64le' or 'float64be' or 'int64' or 'uint64' or 'int64le' or 'uint64le' or 'int64be' or 'uint64be' or 'int32' or 'uint32' or 'int32le' or 'uint32le' or 'int32be' or 'uint32be' or 'int16' or 'uint16' or 'int16le' or 'uint16le' or 'int16be' or 'uint16be' or 'cfloat32' or 'cfloat32le' or 'cfloat32be' or 'cfloat64' or 'cfloat64le' or 'cfloat64be' or 'int8' or 'uint8' or 'bit'] Specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be, uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit.". Maps to a command-line argument: `-datatype %s` (position: 2).

prs [a boolean] Assume that the DW gradients are specified in the PRS frame (Siemens DICOM only). Maps to a command-line argument: `-prs` (position: 3).

quiet [a boolean] Do not display information messages or progress status. Maps to a command-line argument: `-quiet`.

replace_nan_with_zero [a boolean] Replace all NaN values with zero. Maps to a command-line argument: `-zero` (position: 8).

resample [a float] Apply scaling to the intensity values. Maps to a command-line argument: `-scale %d` (position: 6).

stride [a list of from 3 to 4 items which are an integer] Three to four comma-separated numbers specifying the strides of the output data in memory. The actual strides produced will depend on whether the output image format can support it.. Maps to a command-line argument: `-stride %s` (position: 3).

voxel_dims [a list of from 3 to 3 items which are a float] Three comma-separated numbers giving the size of each voxel in mm. Maps to a command-line argument: `-vox %s` (position: 3).

converted [a pathlike object or string representing an existing file] Path/name of 4D volume in voxel order.

MRCrop

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `mrcrop`.

Crops a NIFTI image using the `mrcrop` tool.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> mrcrop = mrt.MRCrop()
>>> mrcrop.inputs.in_file = 'sub-01_dwi.nii.gz'
>>> mrcrop.inputs.in_mask_file = 'sub-01_mod-dwi_desc-brain_mask.nii.gz'
>>> mrcrop.inputs.out_filename = 'sub-01_desc-cropped_dwi.nii.gz'
>>> mrcrop.run()
```

in_file [a pathlike object or string representing an existing file] Input image. Maps to a command-line argument: `%s` (position: -2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

debug [a boolean] Display debugging messages. Maps to a command-line argument: `-debug` (position: 1).

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

in_mask_file [a pathlike object or string representing an existing file] Input mask. Maps to a command-line argument: `-mask %s` (position: -3).

out_filename [a pathlike object or string representing a file] Output cropped image. Maps to a command-line argument: `%s` (position: -1).

quiet [a boolean] Do not display information messages or progress status. Maps to a command-line argument: `-quiet` (position: 1).

cropped [a pathlike object or string representing an existing file] The output cropped image.

MRThreshold

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `mrthreshold`.

Threshold an image using the `mrthreshold` tool.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> mrthresh = mrt.MRCrop()
>>> mrthresh.inputs.in_file = 'sub-01_dwi.nii.gz'
>>> mrthresh.inputs.out_file = 'sub-01_desc-thresholded_dwi.nii.gz'
>>> mrthresh.run()
```

in_file [a pathlike object or string representing an existing file] The input image to be thresholded. Maps to a command-line argument: `%s` (position: -3).

out_file [a pathlike object or string representing a file]

the output binary image mask.

Maps to a command-line argument: `%s` (position: -2).

abs_value [a float] Specify threshold value as absolute intensity. Maps to a command-line argument: `-abs %s` (position: -1).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

force_writing [a boolean] Force file overwriting. Maps to a command-line argument: `-force`.

quiet [a boolean] Do not display information messages or progress status. Maps to a command-line argument: `-quiet`.

thresholded [a pathlike object or string representing an existing file] Path/name of the output binary image mask.

MRTransform

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `mrtransform`.

Apply spatial transformations or reslice images using the `mrtransform` tool.

Example

```
>>> from cmtklib.interfaces.mrtrix3 import MRTransform
>>> MRxform = MRTransform()
>>> MRxform.inputs.in_files = 'anat_coreg.mif'
>>> MRxform.inputs.interp = 'cubic'
>>> MRxform.run()
```

in_files [a list of items which are any value] Input images to be transformed. Maps to a command-line argument: `%s` (position: -2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

debug [a boolean] Display debugging messages. Maps to a command-line argument: `-debug` (position: 1).

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

flip_x [a boolean] Assume the transform is supplied assuming a coordinate system with the x-axis reversed relative to the MRtrix convention (i.e. x increases from right to left). This is required to handle transform matrices produced by FSL's FLIRT command. This is only used in conjunction with the -reference option. Maps to a command-line argument: `-flipx` (position: 1).

interp ['nearest' or 'linear' or 'cubic' or 'sinc'] Set the interpolation method to use when reslicing (choices: nearest,linear, cubic, sinc. Default: cubic). Maps to a command-line argument: `-interp %s`.

invert [a boolean] Invert the specified transform before using it. Maps to a command-line argument: `-inverse` (position: 1).

out_filename [a pathlike object or string representing a file] Output image. Maps to a command-line argument: `%s` (position: -1).

quiet [a boolean] Do not display information messages or progress status. Maps to a command-line argument: `-quiet` (position: 1).

reference_image [a pathlike object or string representing an existing file] In case the transform supplied maps from the input image onto a reference image, use this option to specify the reference. Note that this implicitly sets the -replace option. Maps to a command-line argument: `-reference %s` (position: 1).

replace_transform [a boolean] Replace the current transform by that specified, rather than applying it to the current transform. Maps to a command-line argument: `-replace` (position: 1).

template_image [a pathlike object or string representing an existing file] Reslice the input image to match the specified template image. Maps to a command-line argument: `-template %s` (position: 1).

transformation_file [a pathlike object or string representing an existing file] The transform to apply, in the form of a 4x4 ascii file. Maps to a command-line argument: `-transform %s` (position: 1).

out_file [a pathlike object or string representing an existing file] The output image of the transformation.

MRTrix3Base

Link to code

Bases: nipype.interfaces.base.core.CommandLine

"MRtrix3Base base class inherited by FilterTractogram class.

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

MRtrix_mul

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `mrcalc`.

Multiply two images together using `mrcalc` tool.

Examples

```
>>> from cmtklib.interfaces.mrtrix3 import MRtrix_mul
>>> multiply = MRtrix_mul()
>>> multiply.inputs.input1  = 'image1.nii.gz'
>>> multiply.inputs.input2  = 'image2.nii.gz'
>>> multiply.inputs.out_filename = 'result.nii.gz'
>>> multiply.run()
```

input1 [a pathlike object or string representing an existing file] Input1 file. Maps to a command-line argument: `%s` (position: 1).

input2 [a pathlike object or string representing an existing file] Input2 file. Maps to a command-line argument: `%s` (position: 2).

out_filename [a string] Out filename. Maps to a command-line argument: `-mult %s` (position: 3).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

out_file [a pathlike object or string representing a file] Multiplication result file.

SIFT2

Link to code

Bases: *MRTrix3Base*

Wrapped executable: `tcksift2`.

Determine an appropriate cross-sectional area multiplier for each streamline using `tcksift2` [Smith2015SIFT2].

References

Example

```
>>> import cmtklib.interfaces.mrtrix3 as cmp_mrt
>>> mrtrix_sift2 = cmp_mrt.SIFT2()
>>> mrtrix_sift2.inputs.in_tracks = 'tractogram.tck'
>>> mrtrix_sift2.inputs.in_fod = 'spherical_harmonics_image.nii.gz'
>>> mrtrix_sift2.inputs.out_file = 'sift2_fiber_weights.txt'
>>> mrtrix_sift2.run()
```

in_fod [a pathlike object or string representing an existing file] Input image containing the spherical harmonics of the fibre orientation distributions. Maps to a command-line argument: `%s` (position: -2).

in_tracks [a pathlike object or string representing an existing file] Input track file in TCK format. Maps to a command-line argument: `%s` (position: -3).

act_file [a pathlike object or string representing an existing file] ACT 5TT image file. Maps to a command-line argument: `-act %s` (position: -4).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

out_file [a pathlike object or string representing a file] Output filtered tractogram. Maps to a command-line argument: `%s` (position: -1).

out_tracks [a pathlike object or string representing an existing file] Output filtered tractogram.

StreamlineTrack

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `tckgen`.

Performs tractography using `tckgen`.

It can use one of the following models:

```
'dt_prob', 'dt_stream', 'sd_prob', 'sd_stream'
```

where 'dt' stands for diffusion tensor, 'sd' stands for spherical deconvolution, and 'prob' stands for probabilistic.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> strack = mrt.StreamlineTrack()
>>> strack.inputs.inputmodel = 'SD_PROB'
>>> strack.inputs.in_file = 'data.Bfloat'
>>> strack.inputs.seed_file = 'seed_mask.nii'
>>> strack.run()
```

in_file [a pathlike object or string representing an existing file] The image containing the source data.The type of data required depends on the type of tracking as set in the preceeding argument.For DT methods, the base DWI are needed.For SD methods, the SH harmonic coefficients of the FOD are needed. Maps to a command-line argument: `%s` (position: 2).

act_file [a pathlike object or string representing an existing file] Use the Anatomically-Constrained Tractography framework during tracking; provided image must be in the 5TT (five - tissue - type) format. Maps to a command-line argument: `-act %s`.

angle [a float] Set the maximum angle between successive steps (default is 90deg x stepsize / voxelsize). Maps to a command-line argument: `-angle %s`.

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

backtrack [a boolean] Allow tracks to be truncated. Maps to a command-line argument: `-backtrack`.

crop_at_gmwmi [a boolean] Crop streamline endpoints more precisely as they cross the GM-WM interface. Maps to a command-line argument: `-crop_at_gmwmi`.

cutoff_value [a float] Set the FA or FOD amplitude cutoff for terminating tracks (default is 0.5). Maps to a command-line argument: `-cutoff %s`.

desired_number_of_tracks [an integer] Sets the desired number of tracks.The program will continue to generate tracks until this number of tracks have been selectedand written to the output file (default is 100 for *_STREAM methods, 1000 for *_PROB methods). Maps to a command-line argument: `-select %d`.

do_not_precompute [a boolean] Turns off precomputation of the legendre polynomial values.Warning: this will slow down the algorithm by a factor of approximately 4. Maps to a command-line argument: `-noprecomputed`.

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

gradient_encoding_file [a pathlike object or string representing an existing file] Gradient encoding, supplied as a 4xN text file with each line is in the format [X Y Z b]where [X Y Z] describe the direction of the applied gradient, and b gives the b-valuein units (1000 s/mm^2). See FSL2MRTrix. Maps to a command-line argument: `-grad %s`.

initial_cutoff_value [a float] Sets the minimum FA or FOD amplitude for initiating tracks (default is twice the normal cutoff). Maps to a command-line argument: `-seed_cutoff %s`.

initial_direction [a list of from 2 to 2 items which are an integer] Specify the initial tracking direction as a vector. Maps to a command-line argument: `-seed_direction %s`.

inputmodel ['FACT' or 'iFOD1' or 'iFOD2' or 'Nulldist1' or 'Nulldist2' or 'SD_Stream' or 'Seedtest' or 'Tensor_Det' or 'Tensor_Prob'] Specify the tractography algorithm to use. Valid choices are:FACT, iFOD1, iFOD2, Nulldist1, Nulldist2, SD_Stream, Seedtest, Tensor_Det, Tensor_Prob (default: iFOD2). Maps to a command-line argument: `-algorithm %s` (position: -3). (Nipype **default** value: `FACT`)

mask_file [a pathlike object or string representing an existing file] Mask file. Only tracks within mask. Maps to a command-line argument: `-mask %s`.

maximum_number_of_seeds [an integer] Sets the maximum number of tracks to generate.The program will not generate more tracks than this number,even if the desired number of tracks hasn't yet been reached(default is 1000 x number of streamlines). Maps to a command-line argument: `-seeds %d`.

maximum_tract_length [a float] Sets the maximum length of any track in millimeters (default is 500 mm). Maps to a command-line argument: `-maxlength %s`.

minimum_tract_length [a float] Sets the minimum length of any track in millimeters (default is 5 mm). Maps to a command-line argument: `-minlength %s`.

out_file [a pathlike object or string representing a file] Output data file. Maps to a command-line argument: `%s` (position: -1).

rk4 [a boolean] Use 4th-order Runge-Kutta integration (slower, but eliminates curvature overshoot in 1st-order deterministic methods). Maps to a command-line argument: `-rk4`.

seed_file [a pathlike object or string representing an existing file] Seed file. Maps to a command-line argument: `-seed_image %s`.

seed_gmwmi [a pathlike object or string representing an existing file] Seed from the grey matter - white matter interface (only valid if using ACT framework). Maps to a command-line argument: `-seed_gmwmi %s`. **Requires** inputs: `act_file`.

seed_spec [a list of from 4 to 4 items which are an integer] Seed specification in voxels and radius (x y z r). Maps to a command-line argument: `-seed_sphere %s`.

step_size [a float] Set the step size of the algorithm in mm (default is 0.5). Maps to a command-line argument: `-step %s`.

stop [a boolean] Stop track as soon as it enters any of the include regions. Maps to a command-line argument: `-stop`.

unidirectional [a boolean] Track from the seed point in one direction only (default is to track in both directions). Maps to a command-line argument: `-seed_unidirectional`.

tracked [a pathlike object or string representing an existing file] Output file containing reconstructed tracts.

Tensor2Vector

Link to code

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: `tensor2metric`.

Generates a map of the major eigenvectors of the tensors in each voxel using `tensor2metric`.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> tensor2vector = mrt.Tensor2Vector()
>>> tensor2vector.inputs.in_file = 'dwi_tensor.mif'
>>> tensor2vector.run()
```

in_file [a pathlike object or string representing an existing file] Diffusion tensor image. Maps to a command-line argument: `%s` (position: -2).

args [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

debug [a boolean] Display debugging messages. Maps to a command-line argument: `-debug` (position: 1).

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

out_filename [a pathlike object or string representing a file] Output vector filename. Maps to a command-line argument: `-vector %s` (position: -1).

quiet [a boolean] Do not display information messages or progress status. Maps to a command-line argument: `-quiet` (position: 1).

vector [a pathlike object or string representing an existing file] The output image of the major eigenvectors of the diffusion tensor image.

cmtklib.interfaces.pycartool module

The PyCartool module provides Nipype interfaces with Cartool using pycartool.

CartoolInverseSolutionROIExtraction

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Use Pycartool to load inverse solutions estimated by Cartool.

Examples

```
>>> from cmtklib.interfaces.pycartool import␣
↪CartoolInverseSolutionROIExtraction
>>> cartool_inv_sol = CartoolInverseSolutionROIExtraction()
>>> cartool_inv_sol.inputs.epochs_file = 'sub-01_task-faces_desc-preproc_eeg.
↪set'
>>> cartool_inv_sol.inputs.invsol_file = 'sub-01_eeg.LORETA.is'
>>> cartool_inv_sol.inputs.mapping_spi_rois_file = 'sub-01_atlas-L2018_res-
↪scale1.pickle.rois'
>>> cartool_inv_sol.inputs.lamd = 6
>>> cartool_inv_sol.inputs.svd_toi_begin = 0
>>> cartool_inv_sol.inputs.svd_toi_end = 0.25
>>> cartool_inv_sol.inputs.out_roi_ts_fname_prefix = 'sub-01_task-faces_atlas-
↪L2008_res-scale1_rec-LORETA_timeseries'
>>> cartool_inv_sol.run()
```

References

- https://pycartool.readthedocs.io/en/latest/pycartool.io.html#module-pycartool.io.inverse_solution

epochs_file [a string or os.PathLike object referring to an existing file] Eeg * epochs in .set format.

invsol_file [a string or os.PathLike object referring to an existing file] Inverse solution (.is file loaded with pycartool).

mapping_spi_rois_file [a string or os.PathLike object referring to an existing file] Cartool-reconstructed sources / parcellation ROI mapping file, loaded with pickle.

out_roi_ts_fname_prefix [a string] Output name prefix (no extension) for rois * time series files.

lamb [an integer] Regularization weight.

svd_toi_begin [a float] Start TOI for SVD projection.

svd_toi_end [a float] End TOI for SVD projection.

roi_ts_mat_file [a string or os.PathLike object] Path to output ROI time series file in .mat format.

roi_ts_npy_file [a string or os.PathLike object] Path to output ROI time series file in .npy format.

`static CartoolInverseSolutionROIExtraction.`**`apply_inverse_epochs_cartool`**(*epochs_file*, *invsol_file*, *lamda*, *rois_file*, *svd_params*)

CreateSpiRoisMapping

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Create Cartool-reconstructed sources / parcellation ROI mapping file.

Examples

```
>>> from cmtklib.interfaces.pycartool import CreateSpiRoisMapping
>>> createrois = CreateSpiRoisMapping()
>>> createrois.inputs.roi_volume_file = '/path/to/sub-01_atlas-L2018_res-
↪scale1_dseg.nii.gz'
>>> createrois.inputs.spi_file = '/path/to/sub-01_eeg.spi'
>>> createrois.inputs.out_mapping_spi_rois_fname = 'sub-01_atlas-L2018_res-
↪scale1_eeg.pickle.rois'
>>> createrois.run()
```

out_mapping_spi_rois_fname [a string] Name of output sources / parcellation ROI mapping file in .pickle.rois format.

roi_volume_file [a string or os.PathLike object] Parcellation file in nifti format.

spi_file [a string or os.PathLike object] Cartool reconstructed sources file in spi format.

mapping_spi_rois_file [a string or os.PathLike object] Path to output Cartool-reconstructed sources / parcellation ROI mapping file in .pickle.rois format.

Submodules

cmtklib.carbonfootprint module

cmtklib.config module

Module that defines methods for handling CMP3 configuration files.

`cmtklib.config.`**`anat_load_config_json`**(*pipeline*, *config_path*)

Load the JSON configuration file of an anatomical pipeline.

Parameters

- **pipeline** (*Instance(*`cmp.pipelines.anatomical.anatomical.AnatomicalPipeline`*)*) – Instance of AnatomicalPipeline
- **config_path** (*string*) – Path of the JSON configuration file

`cmtklib.config.`**`anat_save_config`**(*pipeline*, *config_path*)

Save the configuration file of an anatomical pipeline.

Parameters

- **pipeline** (*Instance(*`cmp.pipelines.anatomical.anatomical.AnatomicalPipeline`*)*) – Instance of AnatomicalPipeline
- **config_path** (*string*) – Path of the JSON configuration file

cmtklib.config.**check_configuration_format**(*config_path*)

Check format of the configuration file.

Parameters **config_path** (*string*) – Path to pipeline configuration file

Returns **ext** – Format extension of the pipeline configuration file

Return type '.ini' or '.json'

cmtklib.config.**check_configuration_version**(*config*)

Check the version of CMP3 used to generate a configuration.

Parameters **config** (*Dict*) – Dictionary of configuration parameters loaded from JSON file

Returns **is_same** – True if the version used to generate the configuration matches the version currently used (cmp.info.__version__).

Return type bool

cmtklib.config.**convert_config_ini_2_json**(*config_ini_path*)

Convert a configuration file in old INI format to new JSON format.

Parameters **config_ini_path** (*string*) – Path to configuration file in old INI format

Returns **config_json_path** – Path to converted configuration file in new JSON format

Return type string

cmtklib.config.**create_configparser_from_pipeline**(*pipeline*, *debug=False*)

Create a ConfigParser object from a Pipeline instance.

Parameters

- **pipeline** (*Instance(Pipeline)*) – Instance of pipeline
- **debug** (*bool*) – If True, show additional prints

Returns **config** – Instance of ConfigParser

Return type Instance(configparser.ConfigParser)

cmtklib.config.**create_subject_configuration_from_ref**(*project*, *ref_conf_file*, *pipeline_type*, *multiproc_number_of_cores=1*)

Create the pipeline configuration file for an individual subject from a reference given as input.

Parameters

- **project** (*cmp.project.ProjectInfo*) – Instance of cmp.project.CMP_Project_Info
- **ref_conf_file** (*string*) – Reference configuration file
- **pipeline_type** (*'anatomical', 'diffusion', 'fMRI', 'EEG'*) – Type of pipeline
- **multiproc_number_of_cores** (*int*) – Number of threads used by Nipype

Returns **subject_conf_file** – Configuration file of the individual subject

Return type string

cmtklib.config.**dmri_load_config_json**(*pipeline*, *config_path*)

Load the JSON configuration file of a diffusion pipeline.

Parameters

- **pipeline** (*Instance(cmp.pipelines.diffusion.diffusion.DiffusionPipeline)*) – Instance of DiffusionPipeline

- **config_path** (*string*) – Path of the JSON configuration file

cmtklib.config.**dmri_save_config**(*pipeline*, *config_path*)

Save the INI configuration file of a diffusion pipeline.

Parameters

- **pipeline** (*Instance(cmp.pipelines.diffusion.diffusion.DiffusionPipeline)*) – Instance of DiffusionPipeline
- **config_path** (*string*) – Path of the JSON configuration file

cmtklib.config.**eeg_load_config_json**(*pipeline*, *config_path*)

Load the JSON configuration file of an EEG pipeline.

Parameters

- **pipeline** (*Instance(cmp.pipelines.functional.eeg.EEGPipeline)*) – Instance of EEGPipeline
- **config_path** (*string*) – Path of the JSON configuration file

cmtklib.config.**eeg_save_config**(*pipeline*, *config_path*)

Save the JSON configuration file of a eeg pipeline.

Parameters

- **pipeline** (*Instance(cmp.pipelines.functional.eeg.EEGPipeline)*) – Instance of EEGPipeline
- **config_path** (*string*) – Path of the JSON configuration file

cmtklib.config.**fmri_load_config_json**(*pipeline*, *config_path*)

Load the JSON configuration file of a fMRI pipeline.

Parameters

- **pipeline** (*Instance(cmp.pipelines.functional.fMRI.fMRIPipeline)*) – Instance of fMRIPipeline
- **config_path** (*string*) – Path of the JSON configuration file

cmtklib.config.**fmri_save_config**(*pipeline*, *config_path*)

Save the INI configuration file of a fMRI pipeline.

Parameters

- **pipeline** (*Instance(cmp.pipelines.functional.fMRI.fMRIPipeline)*) – Instance of fMRIPipeline
- **config_path** (*string*) – Path of the JSON configuration file

cmtklib.config.**get_anat_process_detail_json**(*project_info*, *section*, *detail*)

Get the value for a parameter key (detail) in the stage section of the anatomical JSON config file.

Parameters

- **project_info** (*Instance(cmp.project.ProjectInfo)*) – Instance of cmp.project.ProjectInfo class
- **section** (*string*) – Stage section name
- **detail** (*string*) – Parameter key

Returns

Return type The parameter value

cmtklib.config.**get_dmri_process_detail_json**(*project_info*, *section*, *detail*)
Get the value for a parameter key (detail) in the stage section of the diffusion JSON config file.

Parameters

- **project_info** (*Instance(cmp.project.ProjectInfo)*) – Instance of `cmp.project.ProjectInfo` class
- **section** (*string*) – Stage section name
- **detail** (*string*) – Parameter key

Returns

Return type The parameter value

cmtklib.config.**get_eeg_process_detail_json**(*project_info*, *section*, *detail*)
Get the value for a parameter key (detail) in the stage section of the EEG JSON config file.

Parameters

- **project_info** (*Instance(cmp.project.CMP_Project_Info)*) – Instance of `cmp.project.CMP_Project_Info` class
- **section** (*string*) – Stage section name
- **detail** (*string*) – Parameter key

Returns

Return type The parameter value

cmtklib.config.**get_fmri_process_detail_json**(*project_info*, *section*, *detail*)
Get the value for a parameter key (detail) in the stage section of the fMRI JSON config file.

Parameters

- **project_info** (*Instance(cmp.project.ProjectInfo)*) – Instance of `cmp.project.ProjectInfo` class
- **section** (*string*) – Stage section name
- **detail** (*string*) – Parameter key

Returns

Return type The parameter value

cmtklib.config.**get_process_detail_json**(*project_info*, *section*, *detail*)
Get the value for a parameter key (detail) in the global section of the JSON config file.

Parameters

- **project_info** (*Instance(cmp.project.ProjectInfo)*) – Instance of `cmp.project.ProjectInfo` class
- **section** (*string*) – Stage section name
- **detail** (*string*) – Parameter key

Returns

Return type The parameter value

cmtklib.config.**save_configparser_as_json**(*config*, *config_json_path*, *ini_mode=False*, *debug=False*)
Save a ConfigParser to JSON file.

Parameters

- **config** (*Instance(configparser.ConfigParser)*) – Instance of ConfigParser
- **config_json_path** (*string*) – Output path of JSON configuration file
- **ini_mode** (*bool*) – If True, handles all content stored in strings
- **debug** (*bool*) – If True, show additional prints

cmtklib.config.**set_pipeline_attributes_from_config**(*pipeline*, *config*, *debug=False*)
Set the pipeline stage attributes given a configuration.

Parameters

- **pipeline** (*Instance(Pipeline)*) – Instance of pipeline
- **config** (*Dict*) – Dictionary of configuration parameter loaded from the JSON configuration file
- **debug** (*bool*) – If True, show additional prints

cmtklib.connectome module

Module that defines CMTK functions and Nipype interfaces for connectome mapping.

DmriCmat

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Creates the structural connectivity matrices for a given parcellation scheme.

Examples

```
>>> from cmtklib.connectome import DmriCmat
>>> cmat = DmriCmat()
>>> cmat.inputs.base_dir = '/my_directory'
>>> cmat.inputs.track_file = '/path/to/sub-01_tractogram.trk'
>>> cmat.inputs.roi_volumes = ['/path/to/sub-01_space-DWI_atlas-L2018_desc-
↪scale1_dseg.nii.gz',
>>>                             '/path/to/sub-01_space-DWI_atlas-L2018_desc-
↪scale2_dseg.nii.gz',
>>>                             '/path/to/sub-01_space-DWI_atlas-L2018_desc-
↪scale3_dseg.nii.gz',
>>>                             '/path/to/sub-01_space-DWI_atlas-L2018_desc-
↪scale4_dseg.nii.gz',
>>>                             '/path/to/sub-01_space-DWI_atlas-L2018_desc-
↪scale5_dseg.nii.gz']
>>> cmat.inputs.roi_graphmls = ['/path/to/sub-01_atlas-L2018_desc-scale1_dseg.
↪graphml',
>>>                              '/path/to/sub-01_atlas-L2018_desc-scale2_dseg.
↪graphml',
>>>                              '/path/to/sub-01_atlas-L2018_desc-scale3_dseg.
↪graphml',
>>>                              '/path/to/sub-01_atlas-L2018_desc-scale4_dseg.
↪graphml',
```

(continues on next page)

(continued from previous page)

```
>>>                         '/path/to/sub-01_atlas-L2018_desc-scale5_dseg.
↪graphml']
>>> cmat.inputs.parcellation scheme = 'Lausanne2018'
>>> cmat.inputs.output_types = ['gpickle','mat','graphml']
>>> cmat.run()
```

track_file [a list of items which are a pathlike object or string representing an existing file] Tractography result.

additional_maps [a list of items which are a pathlike object or string representing a file] Additional calculated maps (ADC, gFA, . . .).

atlas_info [a dictionary with keys which are any value and with values which are any value] Custom atlas information.

compute_curvature [a boolean] Compute curvature. (Nipype **default** value: `True`)

output_types [a list of items which are a string] Output types of the connectivity matrices.

parcellation_scheme ['Lausanne2018' or 'NativeFreesurfer' or 'Custom'] Parcellation scheme. (Nipype **default** value: `Lausanne2018`)

roi_graphmls [a list of items which are a pathlike object or string representing an existing file] GraphML description of ROI volumes (Lausanne2018).

roi_volumes [a list of items which are a pathlike object or string representing an existing file] ROI volumes registered to diffusion space.

voxel_connectivity [a list of items which are a pathlike object or string representing an existing file] ProbtrackX connectivity matrices (# seed voxels x # target ROIs).

connectivity_matrices [a list of items which are a pathlike object or string representing a file] Connectivity matrices.

endpoints_file [a pathlike object or string representing a file] Numpy files storing the list of fiber endpoint.

endpoints_mm_file [a pathlike object or string representing a file] Numpy files storing the list of fiber endpoint in mm.

filtered_fiberslabel_files [a list of items which are a pathlike object or string representing a file] List of fiber start end ROI parcellation label after filtering.

final_fiberlabels_files [a list of items which are a pathlike object or string representing a file] List of fiber start end ROI parcellation label.

final_fiberslength_files [a list of items which are a pathlike object or string representing a file] List of fiber length.

streamline_final_file [a pathlike object or string representing a file] Final tractogram of fibers considered in the creation of connectivity matrices.

RsfmriCmat

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Creates the functional connectivity matrices for a given parcellation scheme.

It applies scrubbing (if enabled), computes the average GM ROI time-series and computes the
Pearson's correlation coefficient between each GM ROI time-series poir.

Examples

```
>>> from cmtklib.connectome import RsfmriCmat
>>> cmat = RsfmriCmat()
>>> cmat.inputs.base_dir = '/my_directory'
>>> cmat.inputs.func_file = '/path/to/sub-01_task-rest_desc-preproc_bold.nii.gz
↪'
>>> cmat.inputs.roi_volumes = ['/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale1_dseg.nii.gz',
>>>                             '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale2_dseg.nii.gz',
>>>                             '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale3_dseg.nii.gz',
>>>                             '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale4_dseg.nii.gz',
>>>                             '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale5_dseg.nii.gz']
>>> cmat.inputs.roi_graphmls = ['/path/to/sub-01_atlas-L2018_desc-scale1_dseg.
↪graphml',
>>>                              '/path/to/sub-01_atlas-L2018_desc-scale2_dseg.
↪graphml',
>>>                              '/path/to/sub-01_atlas-L2018_desc-scale3_dseg.
↪graphml',
>>>                              '/path/to/sub-01_atlas-L2018_desc-scale4_dseg.
↪graphml',
>>>                              '/path/to/sub-01_atlas-L2018_desc-scale5_dseg.
↪graphml']
>>> cmat.inputs.parcellation scheme = 'Lausanne2018'
>>> cmat.inputs.apply_scrubbing = False
>>> cmat.inputs.output_types = ['gpickle','mat','graphml']
>>> cmat.run()
```

func_file [a pathlike object or string representing an existing file] FMRI volume.

DVARS [a pathlike object or string representing an existing file] DVARS file if scrubbing is performed.

DVARS_th [a float] DVARS threshold.

FD [a pathlike object or string representing an existing file] FD file if scrubbing is performed.

FD_th [a float] FD threshold.

apply_scrubbing [a boolean] Apply scrubbing.

atlas_info [a dictionary with keys which are any value and with values which are any value] Custom atlas information.

output_types [a list of items which are a string] Output types of the connectivity matrices.

parcellation_scheme ['Lausanne2018' or 'NativeFreesurfer' or 'Custom'] Parcellation scheme. (Nipype **default** value: `Lausanne2018`)

roi_graphmls [a list of items which are a pathlike object or string representing an existing file] GraphML description file for ROI volumes (used only if parcellation_scheme == Lausanne2018).

roi_volumes [a list of items which are a pathlike object or string representing an existing file] ROI volumes registered to functional space.

avg_timeseries [a list of items which are a pathlike object or string representing an existing file] ROI average timeseries.

connectivity_matrices [a list of items which are a pathlike object or string representing an existing file] Functional connectivity matrices.

scrubbed_idx [a pathlike object or string representing an existing file] Scrubbed indices.

`cmtklib.connectome.`**`cmat`**(*intrk*, *roi_volumes=None*, *roi_graphmls=None*, *parcellation_scheme=None*, *compute_curvature=True*, *additional_maps=None*, *output_types=None*, *atlas_info=None*)

Create the connection matrix for each resolution using fibers and ROIs.

Parameters

- **intrk** (*TRK file*) – Reconstructed tractogram
- **roi_volumes** (*list*) – List of parcellation files for a given parcellation scheme
- **roi_graphmls** (*list*) – List of graphmls files that describes parcellation nodes
- **parcellation_scheme** (*['NativeFreesurfer', 'Lausanne2018', 'Custom']*) –
- **compute_curvature** (*Boolean*) –
- **additional_maps** (*dict*) – A dictionary of key/value for each additional map where the value is the path to the map
- **output_types** (*['gpickle','mat','graphml']*) –
- **atlas_info** (*dict*) – Dictionary storing information such as path to files related to a parcellation atlas / scheme.

`cmtklib.connectome.`**`compute_curvature_array`**(*fib*)

Computes the curvature array.

`cmtklib.connectome.`**`create_endpoints_array`**(*fib*, *voxelSize*, *print_info*)

Create the endpoints arrays for each fiber.

Parameters

- **fib** (*the fibers data*) –
- **voxelSize** (*3-tuple*) – It contains the voxel size of the ROI image
- **print_info** (*bool*) – If True, print extra information

Returns

- **(endpoints** (*matrix of size [#fibers, 2, 3] containing for each fiber the*) – index of its first and last point in the voxelSize volume
- **endpointsmm)** (*endpoints in milimeter coordinates*)

`cmtklib.connectome.`**`group_analysis_sconn`**(*output_dir*, *subjects_to_be_analyzed*)

Perform group level analysis of structural connectivity matrices.

`cmtklib.connectome.save_fibers(oldhdr, oldfib, fname, indices)`

Stores a new trackvis file fname using only given indices.

Parameters

- **oldhdr** (*the tractogram header*) – Tractogram header to use as reference
- **oldfib** (*the fibers data*) – Input fibers
- **fname** (*string*) – Output tractogram filename
- **indices** (*list*) – Indices of fibers included

cmtklib.data.parcellation.util module

Module that defines CMTK utility functions for retrieving Lausanne parcellation files.

`cmtklib.data.parcellation.util.get_lausanne2018_parcellation_annot(scale='scale1', hemi='lh')`

Return the path of the Freesurfer `.annot` file corresponding to a specific scale and hemisphere.

Parameters

- **scale** (*{'scale1', 'scale2', 'scale3', 'scale4', 'scale5'}*) – Lausanne 2018 parcellation scale
- **hemi** (*{'lh', 'rh'}*) – Brain hemisphere

Returns **annot_file_path** – Absolute path to the queried `.annot` file

Return type string

`cmtklib.data.parcellation.util.get_lausanne2018_parcellation_mni_coords(scale='scale1')`

Return label regions cut coordinates in MNI space (mm).

Parameters **scale** (*{'scale1', 'scale2', 'scale3', 'scale4', 'scale5'}*) – Lausanne 2018 parcellation scale

Returns **coords** – Label regions cut coordinates in MNI space (mm)

Return type numpy.array

cmtklib.data.parcellation.viz module

Module that defines CMTK utility functions for plotting Lausanne parcellation files.

`cmtklib.data.parcellation.viz.plot_lausanne2018_surface_ctx(roi_values, scale='scale1', cmap='Spectral', save_fig=False, output_dir='./', filename=None, fmt='png')`

Plots a set of values on the cortical surface of a given Lausanne 2018 parcellation scale.

Parameters

- **roi_values** (*numpy array*) – The values to be plotted on the surface. The array should have as many values as regions of interest
- **scale** (*{'scale1', 'scale2', 'scale3', 'scale4', 'scale5'}*) – Scale of the Lausanne 2018 atlas to be used
- **cmap** (*string*) – Colormap to use for plotting, default "Spectral"
- **save_fig** (*bool*) – Whether to save the generated figures, default: `False`

- **output_dir** (*string*) – Directory to save the figure, only used when `save_fig == True`
- **filename** (*string*) – Filename of the saved figure (without the extension), only used when `save_fig == True`
- **fmt** (*string*) – Format the figure is saved (Default: "png", also accepted are "pdf", "svg", and others, depending on the backend used)

cmtklib.eeg module

Module that defines CMTK utility functions for the EEG pipeline.

`cmtklib.eeg.`**`save_eeg_connectome_file`**(*output_dir*, *output_basename*, *con_res*, *roi_labels*, *output_types=None*)

Save a dictionary of connectivity matrices with corresponding keys to the metrics in the multiple formats of CMP3.

Parameters

- **output_dir** (*str*) – Output directory for the connectome file(s)
- **output_basename** (*str*) – Base name for the connectome file(s) i.e., `sub-01_atlas-L20018_res-scale1_conndata-network_connectivity`
- **con_res** (*dict*) – Dictionary of connectivity metric / matrix pairs
- **roi_labels** (*list*) – List of parcellation roi labels extracted from the epo.pkl file generated with MNE
- **output_types** (*['tsv', 'gpickle', 'mat', 'graphml']*) – List of output format in which to save the connectome files. (Default: `None`)

cmtklib.diffusion module

Module that defines CMTK utility functions for the diffusion pipeline.

ExtractPVEsFrom5TT

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Create Partial Volume Estimation maps for CSF, GM, WM tissues from `mrtrix3` 5TT image.

Examples

```
>>> from cmtklib.diffusion import ExtractPVEsFrom5TT
>>> pves = ExtractPVEsFrom5TT()
>>> pves.inputs.in_5tt = 'sub-01_desc-5tt_dseg.nii.gz'
>>> pves.inputs.ref_image = 'sub-01_T1w.nii.gz'
>>> pves.inputs.pve_csf_file = '/path/to/output_csf_pve.nii.gz'
>>> pves.inputs.pve_gm_file = '/path/to/output_gm_pve.nii.gz'
>>> pves.inputs.pve_wm_file = '/path/to/output_wm_pve.nii.gz'
>>> pves.run()
```

in_5tt [a pathlike object or string representing an existing file] Input 5TT (4D) image.

pve_csf_file [a pathlike object or string representing a file] CSF Partial Volume Estimation volume estimated from.

pve_gm_file [a pathlike object or string representing a file] GM Partial Volume Estimation volume estimated from.

pve_wm_file [a pathlike object or string representing a file] WM Partial Volume Estimation volume estimated from.

ref_image [a pathlike object or string representing an existing file] Reference 3D image to be used to save 3D PVE volumes.

partial_volume_files [a list of items which are a pathlike object or string representing a file] CSF/GM/WM Partial Volume Estimation images estimated from.

FlipBvec

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Return a diffusion bvec file with flipped axis as specified by `flipping_axis` input.

Examples

```
>>> from cmtklib.diffusion import FlipBvec
>>> flip_bvec = FlipBvec()
>>> flip_bvec.inputs.bvecs = 'sub-01_dwi.bvecs'
>>> flip_bvec.inputs.flipping_axis = ['x']
>>> flip_bvec.inputs.delimiter = ' '
>>> flip_bvec.inputs.header_lines = 0
>>> flip_bvec.inputs.orientation = 'h'
>>> flip_bvec.run()
```

bvecs [a pathlike object or string representing an existing file] Input diffusion gradient bvec file.

delimiter [a string] Delimiter used in the table.

flipping_axis [a list of items which are any value] List of axis to be flipped.

header_lines [an integer] Line number of table header.

orientation [‘v’ or ‘h’] Orientation of the table.

bvecs_flipped [a pathlike object or string representing an existing file] Output bvec file with flipped axis.

FlipTable

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Flip axis and rewrite a gradient table.

Examples

```
>>> from cmtklib.diffusion import FlipTable
>>> flip_table = FlipTable()
>>> flip_table.inputs.table = 'sub-01_mod-dwi_gradient.txt'
>>> flip_table.inputs.flipping_axis = ['x']
>>> flip_table.inputs.orientation = 'v'
>>> flip_table.inputs.delimiter = ','
>>> flip_table.run()
```

delimiter [a string] Delimiter used in the table.

flipping_axis [a list of items which are any value] List of axis to be flipped.

header_lines [an integer] Line number of table header.

orientation ['v' or 'h'] Orientation of the table.

table [a pathlike object or string representing an existing file] Input diffusion gradient table.

table [a pathlike object or string representing an existing file] Output table with flipped axis.

Tck2Trk

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Convert a tractogram in `mrtrix` TCK format to `trackvis` TRK format.

Examples

```
>>> from cmtklib.diffusion import Tck2Trk
>>> tck_to_trk = Tck2Trk()
>>> tck_to_trk.inputs.in_tracks = 'sub-01_tractogram.tck'
>>> tck_to_trk.inputs.in_image = 'sub-01_desc-preproc_dwi.nii.gz'
>>> tck_to_trk.inputs.out_tracks = 'sub-01_tractogram.trk'
>>> tck_to_trk.run()
```

in_image [a pathlike object or string representing an existing file] Input image used to extract the header.

in_tracks [a pathlike object or string representing an existing file] Input track file in MRtrix .tck format.

out_tracks [a pathlike object or string representing a file] Output track file in Trackvis .trk format.

out_tracks [a pathlike object or string representing an existing file] Output track file in Trackvis .trk format.

UpdateGMWMInterfaceSeeding

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Add extra Lausanne2018 structures to the Gray-matter/White-matter interface for tractography seeding.

Examples

```
>>> from cmtklib.diffusion import UpdateGMWMInterfaceSeeding
>>> update_gmwmi = UpdateGMWMInterfaceSeeding()
>>> update_gmwmi.inputs.in_gmwmi_file = 'sub-01_label-gmwmi_desc-orig_dseg.nii.
↪gz'
>>> update_gmwmi.inputs.out_gmwmi_file = 'sub-01_label-gmwmi_desc-modif_dseg.
↪nii.gz'
>>> update_gmwmi.inputs.in_roi_volumes = ['sub-01_space-DWI_atlas-L2018_desc-
↪scale1_dseg.nii.gz',
>>>                                       'sub-01_space-DWI_atlas-L2018_desc-
↪scale2_dseg.nii.gz',
>>>                                       'sub-01_space-DWI_atlas-L2018_desc-
↪scale3_dseg.nii.gz',
>>>                                       'sub-01_space-DWI_atlas-L2018_desc-
↪scale4_dseg.nii.gz',
>>>                                       'sub-01_space-DWI_atlas-L2018_desc-
↪scale5_dseg.nii.gz']
>>> update_gmwmi.run()
```

in_gmwmi_file [a pathlike object or string representing an existing file] Input GMWM interface image used for streamline seeding.

in_roi_volumes [a list of items which are a pathlike object or string representing an existing file] Input parcellation images.

out_gmwmi_file [a pathlike object or string representing a file] Output GM WM interface used for streamline seeding.

out_gmwmi_file [a pathlike object or string representing an existing file] Output GM WM interface used for streamline seeding.

`cmtklib.diffusion.`**`compute_length_array`**(*trkfile=None*, *streams=None*, *savefname='lengths.npy'*)

Computes the length of the fibers in a tractogram and returns an array of length.

Parameters

- **trkfile** (*TRK file*) – Path to the tractogram in TRK format
- **streams** (*the fibers data*) – The fibers from which we want to compute the length
- **savefname** (*string*) – Output filename to write the length array

Returns **fibers_length** – Array of fiber lengths

Return type numpy.array

`cmtklib.diffusion.`**`filter_fibers`**(*intrk*, *outtrk=''*, *fiber_cutoff_lower=20*, *fiber_cutoff_upper=500*)

Filters a tractogram based on lower / upper cutoffs.

Parameters

- **intrk** (*TRK file*) – Path to a tractogram file in TRK format
- **outtrk** (*TRK file*) – Output path for the filtered tractogram
- **fiber_cutoff_lower** (*int*) – Lower number of fibers cutoff (Default: 20)
- **fiber_cutoff_upper** (*int*) – Upper number of fibers cutoff (Default: 500)

cmtklib.functionalMRI module

Module that defines CMTK Nipype interfaces for the Functional MRI pipeline.

Detrending

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Apply linear, quadratic or cubic detrending on the Functional MRI signal.

Examples

```
>>> from cmtklib.functionalMRI import Detrending
>>> detrend = Detrending()
>>> detrend.inputs.base_dir = '/my_directory'
>>> detrend.inputs.in_file = '/path/to/sub-01_task-rest_desc-preproc_bold.nii.
↪gz'
>>> detrend.inputs.gm_file = ['/path/to/sub-01_space-meanBOLD_atlas-L2018_desc-
↪scale1_dseg.nii.gz',
>>>                           '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale2_dseg.nii.gz',
>>>                           '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale3_dseg.nii.gz',
>>>                           '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale4_dseg.nii.gz',
>>>                           '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale5_dseg.nii.gz']
>>> detrend.inputs.mode = 'quadratic'
>>> detrend.run()
```

in_file [a string or os.PathLike object referring to an existing file] FMRI volume to detrend.

gm_file [a list of items which are a string or os.PathLike object referring to an existing file] ROI files registered to fMRI space.

mode ['linear' or 'quadratic' or 'cubic'] Detrending order.

out_file [a string or os.PathLike object referring to an existing file] Detrended fMRI volume.

DiscardTP

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Discards the n first time frame in functional MRI data.

Examples

```
>>> from cmtklib.functionalMRI import DiscardTP
>>> discard = DiscardTP()
>>> discard.inputs.base_dir = '/my_directory'
>>> discard.inputs.in_file = '/path/to/sub-01_task-rest_desc-preproc_bold.nii.
↪gz'
>>> discard.inputs.n_discard = 5
>>> discard.run()
```

in_file [a string or os.PathLike object referring to an existing file] Input 4D fMRI image.

n_discard [an integer] Number of n first frames to discard.

out_file [a string or os.PathLike object referring to an existing file] Output 4D fMRI image with discarded frames.

NuisanceRegression

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Regress out nuisance signals (WM, CSF, movements) through GLM.

Examples

```
>>> from cmtklib.functionalMRI import NuisanceRegression
>>> nuisance = NuisanceRegression()
>>> nuisance.inputs.base_dir = '/my_directory'
>>> nuisance.inputs.in_file = '/path/to/sub-01_task-rest_desc-preproc_bold.nii.
↪gz'
>>> nuisance.inputs.wm_file = '/path/to/sub-01_task-rest_desc-preproc_bold.nii.
↪gz'
>>> nuisance.inputs.csf_file = '/path/to/sub-01_task-rest_desc-preproc_bold.
↪nii.gz'
>>> nuisance.inputs.motion_file = '/path/to/sub-01_motions.par'
>>> nuisance.inputs.gm_file = ['/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale1_dseg.nii.gz',
>>>                             '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale2_dseg.nii.gz',
>>>                             '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale3_dseg.nii.gz',
>>>                             '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale4_dseg.nii.gz',
```

(continues on next page)

(continued from previous page)

```
>>>                             '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↪desc-scale5_dseg.nii.gz']
>>> nuisance.inputs.global_nuisance = False
>>> nuisance.inputs.csf_nuisance = True
>>> nuisance.inputs.wm_nuisance = True
>>> nuisance.inputs.motion_nuisance = True
>>> nuisance.inputs.nuisance_motion_nb_reg = 36
>>> nuisance.inputs.n_discard = 5
>>> nuisance.run()
```

brainfile [a string or os.PathLike object] Eroded brain mask registered to fMRI space.

csf_file [a string or os.PathLike object] Eroded CSF mask registered to fMRI space.

csf_nuisance [a boolean] If `True` perform CSF nuisance regression.

global_nuisance [a boolean] If `True` perform global nuisance regression.

gm_file [a list of items which are a string or os.PathLike object] GM atlas files registered to fMRI space.

in_file [a string or os.PathLike object referring to an existing file] Input fMRI volume.

motion_file [a string or os.PathLike object] Motion nuisance effect.

motion_nuisance [a boolean] If `True` perform motion nuisance regression.

n_discard [an integer] Number of volumes discarded from the fMRI sequence during preprocessing.

nuisance_motion_nb_reg [an integer] Number of reg to use in motion nuisance regression.

wm_file [a string or os.PathLike object] Eroded WM mask registered to fMRI space.

wm_nuisance [a boolean] If `True` perform WM nuisance regression.

averageCSF_mat [a string or os.PathLike object] Output matrix of CSF regression.

averageCSF_npy [a string or os.PathLike object] Output of CSF regression in `npy` format.

averageGlobal_mat [a string or os.PathLike object] Output matrix of global regression.

averageGlobal_npy [a string or os.PathLike object] Output of global regression in `npy` format.

averageWM_mat [a string or os.PathLike object] Output matrix of WM regression.

averageWM_npy [a string or os.PathLike object] Output of WM regression in `npy` format.

out_file [a string or os.PathLike object referring to an existing file] Output fMRI Volume.

Scrubbing

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Computes scrubbing parameters: `FD` and `DVARS`.

Examples

```
>>> from cmtklib.functionalMRI import Scrubbing
>>> scrub = Scrubbing()
>>> scrub.inputs.base_dir = '/my_directory'
>>> scrub.inputs.in_file = '/path/to/sub-01_task-rest_desc-preproc_bold.nii.gz'
>>> scrub.inputs.gm_file = ['/path/to/sub-01_space-meanBOLD_atlas-L2018_desc-
↪scale1_dseg.nii.gz',
>>>                         '/path/to/sub-01_space-meanBOLD_atlas-L2018_desc-
↪scale2_dseg.nii.gz',
>>>                         '/path/to/sub-01_space-meanBOLD_atlas-L2018_desc-
↪scale3_dseg.nii.gz',
>>>                         '/path/to/sub-01_space-meanBOLD_atlas-L2018_desc-
↪scale4_dseg.nii.gz',
>>>                         '/path/to/sub-01_space-meanBOLD_atlas-L2018_desc-
↪scale5_dseg.nii.gz']
>>> scrub.inputs.wm_mask = '/path/to/sub-01_space-meanBOLD_label-WM_dseg.nii.gz
↪'
>>> scrub.inputs.gm_file = '/path/to/sub-01_space-meanBOLD_label-GM_dseg.nii.gz
↪'
>>> scrub.inputs.mode = 'quadratic'
>>> scrub.run()
```

in_file [a string or os.PathLike object referring to an existing file] FMRI volume to scrubb.

gm_file [a list of items which are a string or os.PathLike object referring to an existing file] ROI volumes registered to fMRI space.

motion_parameters [a string or os.PathLike object referring to an existing file] Motion parameters from preprocessing stage.

wm_mask [a string or os.PathLike object referring to an existing file] WM mask registered to fMRI space.

dvars_mat [a string or os.PathLike object referring to an existing file] DVARS matrix for scrubbing.

dvars_npy [a string or os.PathLike object referring to an existing file] DVARS in .npy format.

fd_mat [a string or os.PathLike object referring to an existing file] FD matrix for scrubbing.

fd_npy [a string or os.PathLike object referring to an existing file] FD in .npy format.

cmtklib.parcellation module

Module that defines CMTK utility functions and Nipype interfaces for anatomical parcellation.

CombineParcellations

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Creates the final parcellation.

It combines the original cortico sub-cortical parcellation with the following extra segmented structures:

- Segmentation of the 8 thalamic nuclei per hemisphere
- Segmentation of 14 hippocampal subfields per hemisphere

- Segmentation of 3 brainstem sub-structures

It also generates by defaults the corresponding (1) description of the nodes in `graphml` format and (2) color lookup tables in FreeSurfer format that can be displayed in `freeview`.

Examples

```
>>> parc_combine = CombineParcellations()
>>> parc_combine.inputs.input_rois = ['/path/to/sub-01_atlas-L2018_desc-scale1_
↪dseg.nii.gz',
>>>                                   '/path/to/sub-01_atlas-L2018_desc-scale2_
↪dseg.nii.gz',
>>>                                   '/path/to/sub-01_atlas-L2018_desc-scale3_
↪dseg.nii.gz',
>>>                                   '/path/to/sub-01_atlas-L2018_desc-scale4_
↪dseg.nii.gz',
>>>                                   '/path/to/sub-01_atlas-L2018_desc-scale5_
↪dseg.nii.gz']
>>> parc_combine.inputs.lh_hippocampal_subfields = '/path/to/lh_hippocampal_
↪subfields.nii.gz'
>>> parc_combine.inputs.rh_hippocampal_subfields = '/path/to/rh_hippocampal_
↪subfields.nii.gz'
>>> parc_combine.inputs.brainstem_structures = '/path/to/brainstem_structures.
↪nii.gz'
>>> parc_combine.inputs.thalamus_nuclei = '/path/to/thalamus_nuclei.nii.gz'
>>> parc_combine.inputs.create_colorLUT = True
>>> parc_combine.inputs.create_graphml = True
>>> parc_combine.inputs.subjects_dir = '/path/to/output_dir/freesurfer')
>>> parc_combine.inputs.subject_id = 'sub-01'
>>> parc_combine.run()
```

brainstem_structures [a pathlike object or string representing a file] Brainstem segmentation file.

create_colorLUT [a boolean] If `True`, create the color lookup table in Freesurfer format.

create_graphml [a boolean] If `True`, create the parcellation node description files in `graphml` format.

input_rois [a list of items which are a pathlike object or string representing an existing file] Input parcellation files.

lh_hippocampal_subfields [a pathlike object or string representing a file] Input hippocampal subfields file for left hemisphere.

rh_hippocampal_subfields [a pathlike object or string representing a file] Input hippocampal subfields file for right hemisphere.

subject_id [a string] Freesurfer subject id.

subjects_dir [a pathlike object or string representing a directory] Freesurfer subjects dir.

thalamus_nuclei [a pathlike object or string representing a file] Thalamic nuclei segmentation file.

verbose_level [1 or 2] Verbose level (1: partial (default) / 2: full).

aparc_aseg [a pathlike object or string representing a file] Modified Freesurfer aparc+aseg file.

colorLUT_files [a list of items which are a pathlike object or string representing an existing file] Color lookup table files in Freesurfer format.

graphML_files [a list of items which are a pathlike object or string representing an existing file] Parcellation node description files in `graphml` format.

output_rois [a list of items which are a pathlike object or string representing an existing file] Output parcellation with all structures combined.

`CombineParcellations.ismember(b)`

ComputeParcellationRoiVolumes

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Computes the volumes of each ROI for each parcellation scale.

Examples

```
>>> compute_vol = ComputeParcellationRoiVolumes()
>>> compute_vol.inputs.roi_volumes = ['/path/to/sub-01_atlas-L2018_desc-scale1_
↪dseg.nii.gz',
>>>                                   '/path/to/sub-01_atlas-L2018_desc-scale2_
↪dseg.nii.gz',
>>>                                   '/path/to/sub-01_atlas-L2018_desc-scale3_
↪dseg.nii.gz',
>>>                                   '/path/to/sub-01_atlas-L2018_desc-scale4_
↪dseg.nii.gz',
>>>                                   '/path/to/sub-01_atlas-L2018_desc-scale5_
↪dseg.nii.gz']
>>> compute_vol.inputs.roi_graphmls = ['/path/to/sub-01_atlas-L2018_desc-
↪scale1_dseg.graphml',
>>>                             '/path/to/sub-01_atlas-L2018_desc-scale2_dseg.
↪graphml',
>>>                             '/path/to/sub-01_atlas-L2018_desc-scale3_dseg.
↪graphml',
>>>                             '/path/to/sub-01_atlas-L2018_desc-scale4_dseg.
↪graphml',
>>>                             '/path/to/sub-01_atlas-L2018_desc-scale5_dseg.
↪graphml']
>>> compute_vol.inputs.parcellation_scheme = ['Lausanne2018']
>>> compute_vol.run()
```

parcellation_scheme ['NativeFreesurfer' or 'Lausanne2018' or 'Custom'] Parcellation scheme. (Nipype **default** value: `Lausanne2018`)

roi_graphMLs [a list of items which are a pathlike object or string representing an existing file] GraphML description of ROI volumes (Lausanne2018).

roi_volumes [a list of items which are a pathlike object or string representing an existing file] ROI volumes registered to diffusion space.

roi_volumes_stats [a list of items which are a pathlike object or string representing a file] TSV files with computed parcellation ROI volumes.

Parcellate

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Subdivides segmented ROI file into smaller subregions.

This interface interfaces with the CMTK parcellation functions available in *cmtklib.parcellation* module for all parcellation resolutions of a given scheme.

Example

```
>>> from cmtklib.parcellation import Parcellate
>>> parcellate = Parcellate()
>>> parcellate.inputs.subjects_dir = '/path/to/output_dir/freesurfer'
>>> parcellate.inputs.subject_id = 'sub-01'
>>> parcellate.inputs.parcellation_scheme = 'Lausanne2018'
>>> parcellate.run()
```

subject_id [a string] Subject ID.

erode_masks [a boolean] If `True` erode the masks.

parcellation_scheme ['Lausanne2018' or 'NativeFreesurfer'] Parcellation scheme. (Nipype **default** value: `Lausanne2018`)

subjects_dir [a pathlike object or string representing a directory] Freesurfer main directory.

T1 [a pathlike object or string representing a file] T1 image file.

aparc_aseg [a pathlike object or string representing a file] APArc+ASeg image file (in native space).

aseg [a pathlike object or string representing a file] ASeg image file (in native space).

brain [a pathlike object or string representing a file] Brain-masked T1 image file.

brain_eroded [a pathlike object or string representing a file] Eroded brain file in original space.

brain_mask [a pathlike object or string representing a file] Brain mask file.

csf_eroded [a pathlike object or string representing a file] Eroded csf file in original space.

csf_mask_file [a pathlike object or string representing a file] Cerebrospinal fluid (CSF) mask file.

gray_matter_mask_file [a pathlike object or string representing a file] Cortical gray matter (GM) mask file.

ribbon_file [a pathlike object or string representing an existing file] Image file detailing the cortical ribbon.

roi_files_in_structural_space [a list of items which are a pathlike object or string representing an existing file] ROI image resliced to the dimensions of the original structural image.

white_matter_mask_file [a pathlike object or string representing a file] White matter (WM) mask file.

wm_eroded [a pathlike object or string representing a file] Eroded wm file in original space.

ParcellateBrainstemStructures

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Parcellates the brainstem sub-structures using Freesurfer [Iglesias2015Brainstem].

References

Examples

```
>>> parc_bstem = ParcellateBrainstemStructures()
>>> parc_bstem.inputs.subjects_dir = '/path/to/derivatives/freesurfer'
>>> parc_bstem.inputs.subject_id = 'sub-01'
>>> parc_bstem.run()
```

subject_id [a string] Subject ID.

subjects_dir [a pathlike object or string representing a directory] Freesurfer main directory.

brainstem_structures [a pathlike object or string representing a file] Parcellated brainstem structures file.

ParcellateHippocampalSubfields

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Parcellates the hippocampal subfields using Freesurfer [Iglesias2015Hippo].

References

Examples

```
>>> parc_hippo = ParcellateHippocampalSubfields()
>>> parc_hippo.inputs.subjects_dir = '/path/to/derivatives/freesurfer'
>>> parc_hippo.inputs.subject_id = 'sub-01'
>>> parc_hippo.run()
```

subject_id [a string] Subject ID.

subjects_dir [a pathlike object or string representing a directory] Freesurfer main directory.

lh_hipposubfields [a pathlike object or string representing a file] Left hemisphere hippocampal subfields file.

rh_hipposubfields [a pathlike object or string representing a file] Right hemisphere hippocampal subfields file.

ParcellateThalamus

Link to code

Bases: nipype.interfaces.base.core.BaseInterface

Parcellates the thalamus into 8 nuclei using an atlas-based method [Najdenovska18].

References

Examples

```
>>> parc_thal = ParcellateThalamus()
>>> parc_thal.inputs.T1w_image = File(mandatory=True, desc='T1w image to be␣
↪parcellated')
>>> parc_thal.inputs.bids_dir = Directory(desc='BIDS root directory')
>>> parc_thal.inputs.subject = '01'
>>> parc_thal.inputs.template_image = '/path/to/atlas/T1w.nii.gz'
>>> parc_thal.inputs.thalamic_nuclei_maps = '/path/to/atlas/nuclei/probability/
↪map.nii.gz'
>>> parc_thal.inputs.subjects_dir = '/path/to/output_dir/freesurfer'
>>> parc_thal.inputs.subject_id = 'sub-01'
>>> parc_thal.inputs.ants_precision_type = 'float'
>>> parc_thal.run()
```

T1w_image [a pathlike object or string representing a file] T1w image to be parcellated.

subject_id [a string] Subject ID.

subjects_dir [a pathlike object or string representing a directory] Freesurfer main directory.

template_image [a pathlike object or string representing a file] Template T1w.

thalamic_nuclei_maps [a pathlike object or string representing a file] Probability maps of thalamic nuclei (4D image) in template space.

ants_precision_type ['double' or 'float'] Precision type used during computation.

bids_dir [a pathlike object or string representing a directory] BIDS root directory.

session [a string] Session id.

subject [a string] Subject id.

inverse_warped_image [a pathlike object or string representing a file] Inverse warped template.

max_prob_registered [a pathlike object or string representing a file] Max probability label image (native).

prob_maps_registered [a pathlike object or string representing a file] Probabilistic map of thalamus nuclei (native).

thalamus_mask [a pathlike object or string representing a file] Thalamus mask.

transform_file [a pathlike object or string representing a file] Transform file.

warp_file [a pathlike object or string representing a file] Deformation file.

warped_image [a pathlike object or string representing a file] Template registered to T1w image (native).

`cmtklib.parcellation.`**`create_T1_and_Brain`**(*subject_id*, *subjects_dir*)
Generates T1, T1 masked and aseg+aparc Freesurfer images in NIFTI format.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output_dir/freesurfer)

cmtklib.parcellation.**create_roi**(*subject_id*, *subjects_dir*, *v=True*)

Iteratively creates the ROI_%s.nii.gz files using the given Lausanne2018 parcellation information from networks.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output_dir/freesurfer)
- **v** (*Boolean*) – Verbose mode

cmtklib.parcellation.**create_wm_mask**(*subject_id*, *subjects_dir*, *v=True*)

Creates the white-matter mask using the Freesurfer ribbon as basis in the Lausanne2018 framework.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output_dir/freesurfer)
- **v** (*Boolean*) – Verbose mode

cmtklib.parcellation.**crop_and_move_WM_and_GM**(*subject_id*, *subjects_dir*)

Convert Freesurfer images back to original native space when NativeFreesurfer parcellation scheme is used.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output_dir/freesurfer)

cmtklib.parcellation.**crop_and_move_datasets**(*subject_id*, *subjects_dir*)

Convert Freesurfer images back to original native space when Lausanne2018 parcellation schemes are used.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output_dir/freesurfer)

cmtklib.parcellation.**erode_mask**(*fsdir*, *mask_file*)

Erodes the mask and saves it the Freesurfer subject directory.

Parameters

- **fsdir** (*string*) – Freesurfer subject directory
- **mask_file** (*string*) – Path to mask file

cmtklib.parcellation.**extract**(*Z*, *shape*, *position*, *fill*)

Extract voxel neighbourhood.

Parameters

- **Z** (*numpy.array*) – The original data
- **shape** (*tuple*) – Tuple containing neighbourhood dimensions

- **position** (*tuple*) – Tuple containing central point indexes
- **fill** (*value*) – Value for the padding of Z

Returns **R** – The output neighbourhood of the specified point in Z

Return type numpy.array

`cmtklib.parcellation.`**`generate_WM_and_GM_mask`**(*subject_id*, *subjects_dir*)

Generates the white-matter and gray-matter masks when NativeFreesurfer parcellation is used.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically `/path/to/output_dir/freesurfer`)

`cmtklib.parcellation.`**`get_parcellation`**(*parcel='NativeFreesurfer'*)

Returns a dictionary containing atlas information.

Note: `atlas_info` often used in the code refers to such a dictionary.

Parameters **parcel** (*parcellation scheme*) – It can be: 'NativeFreesurfer' or 'Lausanne2018'

cmtklib.util module

Module that defines CMTK Utility functions.

class `cmtklib.util.`**`BColors`**

Bases: `object`

Utility class for color unicode.

BOLD = '\x1b[1m'

ENDC = '\x1b[0m'

FAIL = '\x1b[91m'

HEADER = '\x1b[95m'

OKBLUE = '\x1b[94m'

OKGREEN = '\x1b[92m'

UNDERLINE = '\x1b[4m'

WARNING = '\x1b[93m'

`cmtklib.util.`**`check_directory_exists`**(*mandatory_dir*)

Makes sure the mandatory directory exists.

Raises `FileNotFoundError` – Raised when the directory is not found.

`cmtklib.util.`**`convert_list_to_tuple`**(*lists*)

Convert list of files to tuple of files.

(Duplicated with preprocessing, could be moved to utils in the future)

Parameters **lists** (*[bvecs, bvals]*) – List of files containing bvecs and bvals

Returns **out_tuple** – Tuple of files containing bvecs and bvals

Return type (bvecs, bvals)

`cmtklib.util.extract_freesurfer_subject_dir`(*reconall_report*, *local_output_dir=None*, *debug=False*)

Extract Freesurfer subject directory from the report created by Nipype Freesurfer Recon-all node.

Parameters

- **reconall_report** (*string*) – Path to the recon-all report
- **local_output_dir** (*string*) – Local output / derivatives directory
- **debug** (*bool*) – If `True`, show printed outputs

Returns **fs_subject_dir** – Freesurfer subject directory

Return type string

`cmtklib.util.extract_reconall_base_dir`(*file*)

Extract Recon-all base directory from a file.

Parameters **file** (*File*) – File generated by Recon-all

Returns **out_path** – Recon-all base directory

Return type string

`cmtklib.util.find_toolbox_derivatives_containing_file`(*bids_dir*, *fname*, *debug=False*)

Find the toolbox derivatives directory in the derivatives folder of the BIDS dataset containing a file.

This function is used by the EEGPipeline.

Parameters

- **bids_dir** (*str*) – Path the BIDS root directory
- **fname** (*str*) – Filename to find
- **debug** (*bool*) – If `True`, print the directory found

Returns **out_tuple** – Tuple of files containing bvecs and bvals

Return type (bvecs, bvals)

`cmtklib.util.get_basename`(*path*)

Return `os.path.basename()` of a `path`.

Parameters **path** (*os.path*) – Path to extract the containing directory

Returns **path** – Path to the containing directory

Return type os.path

`cmtklib.util.get_freesurfer_subject_id`(*file*)

Extract Freesurfer subject ID from file generated by recon-all.

Parameters **file** (*str*) – File generated by recon-all

Returns **out** – Freesurfer subject ID

Return type str

`cmtklib.util.get_pipeline_dictionary_outputs`(*datasink_report*, *local_output_dir=None*, *debug=False*)

Read the Nipype datasink report and return a dictionary of pipeline outputs.

Parameters

- **datasink_report** (*string*) – Path to the datasink report
- **local_output_dir** (*string*) – Local output / derivatives directory

- **debug** (*bool*) – If True, print output dictionary

Returns dict_outputs – Dictionary of pipeline outputs

Return type dict

`cmtklib.util.isavailable(file)`
Check if file is available and return the file if it is.

Used for debugging.

Parameters file (*File*) – Input file

Returns file – Output file

Return type File

`cmtklib.util.length(xyz, along=False)`
Euclidean length of track line.

Parameters

- **xyz** (*array-like shape (N,3)*) – array representing x,y,z of N points in a track
- **along** (*bool, optional*) – If True, return array giving cumulative length along track, otherwise (default) return scalar giving total length.

Returns L – scalar in case of `along` == False, giving total length, array if `along` == True, giving cumulative lengths.

Return type scalar or array shape (N-1,)

Examples

```
>>> xyz = np.array([[1,1,1],[2,3,4],[0,0,0]])
>>> expected_lens = np.sqrt([1+2**2+3**2, 2**2+3**2+4**2])
>>> length(xyz) == expected_lens.sum()
True
>>> len_along = length(xyz, along=True)
>>> np.allclose(len_along, expected_lens.cumsum())
True
>>> length([])
0
>>> length([[1, 2, 3]])
0
>>> length([], along=True)
array([0])
```

`cmtklib.util.magn(xyz, n=1)`
Returns the vector magnitude

Parameters

- **xyz** (*vector*) – Input vector
- **n** (*int*) – Tile by n if n>1 before return

`cmtklib.util.mean_curvature(xyz)`
Calculates the mean curvature of a curve.

Parameters xyz (*array-like shape (N,3)*) – array representing x,y,z of N points in a curve

Returns **m** – float representing the mean curvature

Return type float

Examples

Create a straight line and a semi-circle and print their mean curvatures

```
>>> from dipy.tracking import metrics as tm
>>> import numpy as np
>>> x=np.linspace(0,1,100)
>>> y=0*x
>>> z=0*x
>>> xyz=np.vstack((x,y,z)).T
>>> m=tm.mean_curvature(xyz)  # mean curvature straight line
>>> theta=np.pi*np.linspace(0,1,100)
>>> x=np.cos(theta)
>>> y=np.sin(theta)
>>> z=0*x
>>> xyz=np.vstack((x,y,z)).T
>>> m=tm.mean_curvature(xyz)  # mean curvature for semi-circle
```

`cmtklib.util.`**`print_blue`**(*message*)

Print blue-colored message

Parameters **`message`** (*`string`*) – The string of the message to be printed

`cmtklib.util.`**`print_error`**(*message*)

Print red-colored error message

Parameters **`message`** (*`string`*) – The string of the message to be printed

`cmtklib.util.`**`print_warning`**(*message*)

Print yellow-colored warning message

Parameters **`message`** (*`string`*) – The string of the message to be printed

`cmtklib.util.`**`return_button_style_sheet`**(*image*, *image_disabled=None*, *verbose=False*)

Return Qt style sheet for QPushButton with image

Parameters

- **`image`** (*`string`*) – Path to image to use as icon when button is enabled
- **`image_disabled`** (*`string`*) – Path to image to use as icon when button is disabled
- **`verbose`** (*`Bool`*) – Print the style sheet if True Default: False

Returns **button_style_sheet** – Qt style sheet for QPushButton with image

Return type string

`cmtklib.util.`**`unicode2str`**(*text*)

Convert a unicode to a string using system's encoding.

Parameters **`text`** (*`bytes`*) – Unicode bytes representation of a string

Returns **out_str** – Output string

Return type str

5.7 Adopting Datalad for collaboration

Datalad is a powerful tool for the versioning and sharing of raw and processed data as well as for the tracking of data provenance (i.e. the recording on how data was processed). This page was created with the intention to share with the user how we adopted datalad to manage and process datasets with Connectome Mapper 3 in our lab, following the YODA principles to our best.

You may ask "What are the YODA principles?". They are basic principles behind creating, sharing, and publishing reproducible, understandable, and open data analysis projects with DataLad.

For more details and tutorials on Datalad and YODA, please check the recent Datalad Handbook and the YODA principles.

Happy Collaborative and Reproducible Connectome Mapping!

5.7.1 Prerequisites

- Python3 must be installed with Datalad and all dependencies. You can use the conda environment `py39cmp-gui` for instance. See *Installation of py39cmp-gui* for more installation details.
- A recent version of `git-annex` and `liblzma` (included in `py39cmp-gui` for Ubuntu/Debian).
- Docker must be installed on systems running Connectome Mapper 3. See *Prerequisites of Connectome Mapper 3* for more installation instructions.

5.7.2 Copy BIDS dataset to server

Copy the raw BIDS dataset using `rsync`:

```
rsync -P -avz -e 'ssh' \
--exclude 'derivatives' \
--exclude 'code' \
--exclude '.datalad' \
--exclude '.git' \
--exclude '.gitattributes' \
/path/to/ds-example/* \
<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-example
```

where:

- `-P` is used to show progress during transfer
- `-v` increases verbosity
- `-e` specifies the remote shell to use (ssh)
- `-a` indicates archive mode
- `-z` enables file data compression during the transfer
- `--exclude DIR_NAME` exclude the specified `DIR_NAME` from the copy

5.7.3 Remote datalad dataset creation on Server

Connect to Server

To connect with SSH:

```
ssh <SERVER_USERNAME>@<SERVER_IP_ADDRESS>
```

Creation of Datalad dataset

Go to the source dataset directory:

```
cd /archive/data/ds-example
```

Initialize the Datalad dataset:

```
datalad create -f -c text2git -D "Original example dataset on lab server" -d .
```

where:

- `-f` forces to create the datalad dataset if not empty
- `-c text2git` configures Datalad to use git to manage text file
- `-D` gives a brief description of the dataset
- `-d` specify the location where the Datalad dataset is created

Track all files contained in the dataset with Datalad:

```
datalad save -m "Source (Origin) BIDS dataset" --version-tag origin
```

where:

- `-m MESSAGE` is the description of the state or the changes made to the dataset
- `--version-tag` tags the state of the Dataset

Report on the state of dataset content:

```
datalad status -r
git log
```

5.7.4 Processing using the Connectome Mapper BIDS App on Alice's workstation

Processed dataset creation

Initialize a datalad dataset with the YODA procedure:

```
datalad create -c text2git -c yoda \
-D "Processed example dataset by Alice with CMP3" \
/home/alice/data/ds-example-processed
```

This will create a datalad dataset with:

- a code directory in your dataset

- three files for human consumption (README.md, CHANGELOG.md)
- everything in the code/ directory configured to be tracked by Git, not git-annex
- README.md and CHANGELOG.md configured in the root of the dataset to be tracked by Git
- Text files configured to be tracked by Git

Go to the created dataset directory:

```
cd /home/alice/data/ds-example-processed
```

Create the derivatives output directory:

```
mkdir derivatives
```

Raw BIDS dataset installation

Install the remove datalad dataset ds-example in /home/alice/data/ds-example-processed/input/:

```
datalad install -d . -s ssh://<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-
↪example \
/home/alice/data/ds-example-processed/input/
```

where:

- -s SOURCE specifies the URL or local path of the installation source

Get T1w and Diffusion images to be processed

For reproducibility, create and write datalad get commands to get_required_files_for_analysis.sh:

```
echo "datalad get input/sub-*/ses-*/anat/sub-*_T1w.nii.gz" > code/get_required_files_for_
↪analysis.sh
echo "datalad get input/sub-*/ses-*/dwi/sub-*_dwi.nii.gz" >> code/get_required_files_for_
↪analysis.sh
echo "datalad get input/sub-*/ses-*/dwi/sub-*_dwi.bvec" >> code/get_required_files_for_
↪analysis.sh
echo "datalad get input/sub-*/ses-*/dwi/sub-*_dwi.bval" >> code/get_required_files_for_
↪analysis.sh
```

Save the script to the dataset's history:

```
datalad save -m "Add script to get the files required for analysis by Alice"
```

Execute the script:

```
sh code/get_required_files_for_analysis.sh
```

Link the container image with the dataset

Add Connectome Mapper's container image to the datalad dataset:

```
datalad containers-add connectomemapper-bidsapp-<VERSION_TAG> \
--url dhub://sebastientourbier/connectomemapper-bidsapp:<VERSION_TAG> \
-d . \
--call-fmt \
"docker run --rm -t \
    -v "$(pwd)/input":/bids_dir \
    -v "$(pwd)/code":/bids_dir/code \
    -v "$(pwd)"/derivatives:/output_dir \
    -u "$(id -u)":"$(id -g)" \
    sebastientourbier/connectomemapper-bidsapp:<VERSION_TAG> {cmd}"
```

Note: `--call-fmt` specifies a custom docker run command. The current directory is assumed to be the BIDS root directory and retrieve with `"$(pwd)"/input` and the output directory is inside the `derivatives/` folder.

Important: The name of the container-name registered to Datalad cannot have dot as character so that a <VERSION_TAG> of `v3.X.Y` would need to be rewritten as `v3-X-Y`

Copy existing reference pipeline configuration files to `code` folder:

```
cp /path/to/existing/ref_anatomical_config.json \
code/ref_anatomical_config.json
cp /path/to/existing/ref_diffusion_config.json \
code/ref_diffusion_config.json
```

Copy FreeSurfer license file to `code` folder:

```
cp /path/to/freesurfer/license.txt \
code/license.txt
```

Save the state of the dataset prior to analysis:

```
datalad save -m "Alice's test dataset on local \
workstation ready for analysis with connectomemapper-bidsapp:<VERSION_TAG>" \
--version-tag ready4analysis-<date>-<time>
```

Run Connectome Mapper with Datalad

Run Connectome Mapper on all subjects:

```
datalad containers-run --container-name connectomemapper-bidsapp-<VERSION_TAG> \
--input code/ref_anatomical_config.json \
--input code/ref_diffusion_config.json \
--output derivatives \
/bids_dir /output_dir participant \
--anat_pipeline_config '/bids_dir/{inputs[0]}' \
--dwi_pipeline_config '/bids_dir/{inputs[1]}'
```

Note: `datalad containers-run` will take of replacing the `{inputs[i]}` by the value specified by the *i* `--input` flag (Indexing start at 0).

Save the state:

```
datalad save -m "Alice's test dataset on local \
workstation processed by connectomemapper-bidsapp:<VERSION_TAG>, {Date/Time}" \
--version-tag processed-<date>-<time>
```

Report on the state of dataset content:

```
datalad status -r
git log
```

Configure a datalad dataset target on the Server

Create a remote dataset repository and configures it as a dataset sibling to be used as a publication target:

```
datalad create-sibling --name remote -d . \
<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-example-processed
```

See the documentation of datalad create-sibling command for more details.

Update the remote datalad dataset

Push the datalad dataset with data derivatives to the server:

```
datalad push -d . --to remote
```

Note: `--to remote` specifies the `remote` dataset sibling i.e. `ssh://<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-example-processed` previously configured.

Uninstall all files accessible from the remote

With DataLad we don't have to keep those inputs around so you can safely uninstall them without losing the ability to reproduce an analysis:

```
datalad uninstall input/sub-*/*
```

5.7.5 Local collaboration with Bob for Electrical Source Imaging

Processed dataset installation on Bob's workstation

Install the processed datalad dataset `ds-example-processed` in `/home/bob/data/ds-example-processed`:

```
datalad install -s ssh://<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-example-
↪processed  \
/home/bob/data/ds-example-processed
```

Go to datalad dataset clone directory:

```
cd /home/bob/data/ds-example-processed
```

Get connectome mapper output files (Brain Segmentation and Multi-scale Parcellation) used by Bob in his analysis

For reproducibility, write datalad get commands to `get_required_files_for_analysis_by_bob.sh`:

```
echo "datalad get derivatives/cmp/sub-*/ses-*/anat/sub-*_mask.nii.gz" \
> code/get_required_files_for_analysis_by_bob.sh
echo "datalad get derivatives/cmp/sub-*/ses-*/anat/sub-*_class-*_dseg.nii.gz" \
>> code/get_required_files_for_analysis_by_bob.sh
echo "datalad get derivatives/cmp/sub-*/ses-*/anat/sub-*_scale*_atlas.nii.gz" \
>> code/get_required_files_for_analysis_by_bob.sh
```

Save the script to the dataset's history:

```
datalad save -m "Add script to get the files required for analysis by Bob"
```

Execute the script:

```
sh code/get_required_files_for_analysis_by_bob.sh
```

Update derivatives

Update derivatives with data produced by Cartool:

```
cd /home/bob/data/ds-example
mkdir derivatives/cartool
cp [...]
```

Save the state:

```
datalad save -m "Bob's test dataset on local \
workstation processed by cartool:<CARTOOL_VERSION>, {Date/Time}" \
--version-tag processed-<date>-<time>
```

Report on the state of dataset content:

```
datalad status -r
git log
```

Update the remote datalad dataset

Update the remote datalad dataset with data derivatives:

```
datalad push -d . --to origin
```

Note: `--to origin` specifies the `origin` dataset sibling i.e. `ssh://<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-example-processed` from which it was cloned.

Uninstall all files accessible from the remote

Again, with DataLad we don't have to keep those inputs around so you can safely uninstall them without losing the ability to reproduce an analysis:

```
datalad uninstall derivatives/cmp-*/*
datalad uninstall derivatives/freesurfer-*/*
datalad uninstall derivatives/nipype-*/*
```

Authors Sebastien Tourbier

Version Revision: 2.1 (Last modification: 2022 Feb 09)

5.8 Running on a cluster (HPC)

Connectome Mapper 3 BIDS App can be run on a cluster using Singularity.

For your convenience, the Singularity image is automatically built along the docker image using Singularity `3.8.4` and deployed to Sylabs.io (equivalent of DockerHub for Singularity) during continuous integration on CircleCI. It can be freely downloaded with the following command:

```
$ singularity pull library://connectomicslab/default/connectomemapper-bidsapp:latest
```

If you prefer, you can still build the Singularity image on your side using one of the 2 methods described in *Conversion to a Singularity image*.

A list of useful singularity command can be found in *Useful singularity commands*. For more documentation about Singularity, please check the official documentation website.

Happy Large-Scale Connectome Mapping!

5.8.1 Prerequisites

- Singularity must be installed. Check the official documentation webpage for installation instructions.

5.8.2 Running the singularity image

The following example shows how to call from the terminal the Singularity image of the CMP3 BIDS App to perform both anatomical and diffusion pipelines for `sub-01`, `sub-02` and `sub-03` of a BIDS dataset whose root directory is located at `${localDir}`:

```
$ singularity run --containall \
        --bind ${localDir}:/bids_dir --bind ${localDir}/derivatives:/output_dir \
            library://connectomicslab/default/connectomemapper-bidsapp:|release| \
            /bids_dir /output_dir participant --participant_label 01 02 03 \
            --anat_pipeline_config /bids_dir/code/ref_anatomical_config.json \
            --dwi_pipeline_config /bids_dir/code/ref_diffusion_config.json \
            --fs_license /bids_dir/code/license.txt \
            --number_of_participants_processed_in_parallel 3
```

Note: As you can see, the `singularity run` command is slightly different from the `docker run`. The docker option flag `-v` is replaced by the singularity `--bind` to map local folders inside the container. Last but not least, while docker containers are executed in total isolation, singularity images MUST run with the option flag `--containall`. Otherwise your $HOME and $TMP directories or your local environment variables might be shared inside the container.

5.8.3 Conversion to a Singularity image

It actually exists two options for Docker to Singularity container image conversion. Let's say we want to store Singularity-compatible image file in `~/Softwares/singularity/`.

Option 1 (recommended): Using the Docker image docker2singularity

1. Build locally in a `/tmp/test` folder:

```
$ mkdir -p /tmp/test
$ docker run -v /var/run/docker.sock:/var/run/docker.sock  \
     -v /tmp/test:/output --privileged -t --rm  \
     singularityware/docker2singularity  \
     --name cmp-v3.1.0.simg  \
     sebastientourbier/connectomemapper-bidsapp:v3.1.0
```

2. Move the converted image `cmp-|release|` to the `~/Softwares/singularity` folder on the cluster (via ssh using scp for instance)

```
$ scp -v /tmp/test/cmp-v3.1.0.simg <user>@<cluster_url>:~/Softwares/singularity/
↪cmp-v3.1.0.simg
```

Advantage(s): Has never failed

Disadvantage(s): Have to make a-priori the conversion locally on a workstation where docker is installed and then upload the converted image to the cluster

Option 2 : Using singularity directly

```
$ singularity build ~/Softwares/singularity/cmp-v3.1.0.simg  \
        docker://sebastientourbier/connectomemapper-bidsapp:v3.1.0
```

This command will directly download the latest version release of the Docker image from the DockerHub and convert it to a Singularity image.

Advantage(s): Can be executed on the cluster directly

Disadvantage(s): Has shown to fail because of some docker / singularity version incompatibilities

5.8.4 Useful singularity commands

- Display a container's metadata:

  ```
  $ singularity inspect ~/Softwares/singularity/cmp-v3.1.0.simg
  ```

- Clean cache:

  ```
  $ singularity cache clean
  ```

Authors Sebastien Tourbier

Version Revision: 2 (Last modification: 2021 Jan 04)

5.9 Tutorial notebooks

5.9.1 Analysis Tutorial

This tutorial demonstrates how to analyze and interpret the outputs from Connectomemapper 3. In particular it will tell you how to:

- Get the list of connectome files with Pybids
- Read the .tsv connectome files with Networkx and Pandas
- Read the .gpickle files with Networkx
- Read the .mat files with Scipy
- Compute the connectome harmonics with PyGSP
- Visualize the harmonics with the plot functions of Nilearn

Setup instructions

If you want to reproduce all the results of this notebook on your side, a conda `environment.yml` file can be downloaded at the following link: tutorial_environment.yml. The original `.ipynb` notebook file can be downloaded at the following link: analysis_tutorial.ipynb.

Once you have downloaded the conda environment file, install the environment as follows:

```
$ conda env create -f /path/to/downloaded/analysis_tutorial.yml
```

This will install all the packages needed to run this notebook including jupyter lab.

You can then activate it, go to the directory where you downloaded the `analysis_tutorial.ipynb`, and launch jupyter lab as follows:

```
$ cd /directory/of/downloaded/analysis_tutorial.ipynb/
$ conda activate cmp3-tutorial
$ jupyter lab
```

You are ready to open and interact with the notebook!

Loading the external python packages

[1]:
```
# General
import os
import sys
import subprocess
import copy

# Dataset management
import datalad.api as dl

# Data handling
import pandas as pd
import numpy as np
import nibabel as nib
import scipy.io as sio

# BIDS dataset handling
from bids import BIDSLayout

# Network / Graph
import pygsp
import networkx as nx

# Visualization
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.colors import LogNorm
import nilearn
from nilearn import plotting, image, datasets
```

```
/Applications/miniconda3/envs/cmp3-tutorial/lib/python3.7/site-packages/nilearn/datasets/
↪__init__.py:96: FutureWarning: Fetchers from the nilearn.datasets module will be␣
↪updated in version 0.9 to return python strings instead of bytes and Pandas dataframes␣
↪instead of Numpy arrays.
  "Numpy arrays.", FutureWarning)
```

Loading the connectome files

For demonstration, we are going to use the latest version of VEPCON dataset, available on Open Neuro that already contains output from Connectome Mapper v3.0.3. A full description of the dataset can be found in Pascucci, Tourbier, et al. 2022.

In case you want to rerun the notebook, make sure to remove any `ds003505_demo` folder in the directory of the notebook. Otherwise, datalad install will complain.

[2]:
```
%%time
# Download example dataset with datalad
dataset_dir = os.path.join(".", "ds003505_demo")
vepcon_data = dl.install(
    path=dataset_dir,
    source="https://github.com/OpenNeuroDatasets/ds003505.git"
)
```

```
[INFO] Cloning dataset to Dataset(/Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo)
[INFO] Attempting to clone from https://github.com/OpenNeuroDatasets/ds003505.git to /
↪Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo
[INFO] Start enumerating objects
[INFO] Start receiving objects
[INFO] Start resolving deltas
[INFO] Completed clone attempts for Dataset(/Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo)
[INFO] scanning for annexed files (this may take some time)
[INFO] Remote origin not usable by git-annex; setting annex-ignore
[INFO] https://github.com/OpenNeuroDatasets/ds003505.git/config download failed: Not␣
↪Found
[INFO] access to 1 dataset sibling s3-PRIVATE not auto-enabled, enable with:
|               datalad siblings -d "/Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo" enable -s s3-PRIVATE
```

```
CPU times: user 97 ms, sys: 74.8 ms, total: 172 ms
Wall time: 9.41 s
```

As the dataset is in BIDS, we can use Pybids to help us with the task of interacting with the files of the dataset.

[3]:
```
# Represent the BIDS dataset as a PyBIDS BIDSLayout
layout = BIDSLayout(vepcon_data.path)
# Add derivative folder containing the connectivity matrices
layout.add_derivatives(os.path.join(vepcon_data.path, "derivatives", "cmp-v3.0.3"))
```

Now we can easily query for the filenames of the files we are interested in using `layout.get`. We will ask for the connectivity matrix of subject 01, scale 3, in tsv format.

It will be returned as a list of file paths (in this case containing just one element). Note that at this stage the Datalad dataset contains mostly file annexes.

As the connectome in TSV format, are just text files, they are directly managed by Git, such that we do have to retrieve its actual content before querying them.

[4]:
```
# Query the connectome file path
conn_tsv_scale3 = layout.get(
```

(continues on next page)

(continued from previous page)

```
    subject='01',
    datatype='dwi',
    atlas='L2018',
    res='scale3',
    suffix='connectivity',
    extension='tsv',
    return_type='filename'
)
conn_tsv_scale3
```

[4]:
```
['/Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/cmp-v3.0.3/sub-01/dwi/sub-01_atlas-L2018_res-scale3_conndata-network_
↪connectivity.tsv']
```

We can then use Pandas to read the file and display it as a table.

[5]:
```
edges = pd.read_csv(conn_tsv_scale3[0], delimiter="\t")
edges.head()
```

[5]:
```
   source  target  number_of_fibers  fiber_length_mean  fiber_length_median  \
0       1       1               193           6.800518             6.000000
1       1       8                23          16.173914            17.000004
2       1       3                98           9.382652             8.000008
3       1       4                 5          23.299999            23.000008
4       1     111                63           9.484127             8.499999

   fiber_length_std  fiber_proportion  fiber_density  \
0          2.242513          0.118146       0.007391
1          5.212046          0.014080       0.000411
2          3.732647          0.059991       0.002899
3          0.678231          0.003061       0.000070
4          3.311797          0.038566       0.001030

   normalized_fiber_density   FA_mean    FA_std  FA_median  ADC_mean  \
0                  0.052092  0.196300  0.084466   0.177124  0.000785
1                  0.002896  0.227573  0.094389   0.218950  0.000815
2                  0.020433  0.195890  0.081233   0.185918  0.000786
3                  0.000490  0.229370  0.093624   0.229142  0.000787
4                  0.007262  0.242134  0.064675   0.254462  0.000816

    ADC_std  ADC_median
0  0.000082    0.000777
1  0.000067    0.000813
2  0.000069    0.000790
3  0.000086    0.000782
4  0.000096    0.000803
```

Using Networkx, we can convert this table to a network graph. From that, we can convert individual measures to a Numpy array. The array format is especially useful, as it allows us to plot the edge weights easily.

[6]:
```
G = nx.from_pandas_edgelist(edges, edge_attr=True)
A_fiber_density = nx.to_numpy_array(G, weight="fiber_density")
A_fiber_density
```

[6]:
```
array([[0.00739067, 0.00041088, 0.00289892, ..., 0.        , 0.        ,
        0.        ],
       [0.00041088, 0.00541092, 0.00340868, ..., 0.        , 0.        ,
        0.        ],
       [0.00289892, 0.00340868, 0.0055656 , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 0.00909844, 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.00863366,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.00504128]])
```

Alternatively, we can also load the matrices in network format, by reading the gpickle files using Networkx:

[7]:
```
# Retrieve content of the connectome file with datalad
vepcon_data.get('derivatives/cmp-v3.0.3/sub-01/dwi/sub-01_atlas-L2018_res-scale3_
↪conndata-network_connectivity.gpickle')
# Query the connectome file path
conn_gpickle_scale3 = layout.get(
    subject='01',
    datatype='dwi',
    atlas='L2018',
    res='scale3',
    suffix='connectivity',
    extension='gpickle',
    return_type='filename'
)
G = nx.read_gpickle(conn_gpickle_scale3[0])  # same format as with tsv
A_fiber_density = nx.to_numpy_array(G, weight="fiber_density")
A_fiber_density
```

[7]:
```
array([[7.39067471e-03, 4.01920257e-04, 2.89891974e-03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [4.01920257e-04, 7.98799252e-03, 6.66710786e-03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [2.89891974e-03, 6.66710786e-03, 5.56559629e-03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        5.84868693e-03, 6.47767105e-04, 1.57817881e-04],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        6.47767105e-04, 1.18190323e-03, 1.05523480e-05],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        1.57817881e-04, 1.05523480e-05, 8.37270426e-05]])
```

…or load the .mat files with Scipy:

[8]:
```
# Retrieve content of the connectome file with datalad
vepcon_data.get('derivatives/cmp-v3.0.3/sub-01/dwi/sub-01_atlas-L2018_res-scale3_
↪conndata-network_connectivity.mat')
# Query the connectome file path
conn_mat_scale3 = layout.get(
```

(continues on next page)

(continued from previous page)

```
    subject='01',
    datatype='dwi',
    atlas='L2018',
    res='scale3',
    suffix='connectivity',
    extension='mat',
    return_type='filename'
)
A_mat = sio.loadmat(conn_mat_scale3[0])
```

The adjacency matrices here can be accessed as followss:

[9]:
```
A_mat["sc"]["fiber_density"][0][0]
```

[9]:
```
array([[7.39067471e-03, 4.01920257e-04, 2.89891974e-03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [4.01920257e-04, 7.98799252e-03, 6.66710786e-03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [2.89891974e-03, 6.66710786e-03, 5.56559629e-03, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       ...,
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        5.84868693e-03, 6.47767105e-04, 1.57817881e-04],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        6.47767105e-04, 1.18190323e-03, 1.05523480e-05],
       [0.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        1.57817881e-04, 1.05523480e-05, 8.37270426e-05]])
```

Note that in these two situations, the connectome files are not directly managed by Git and their actual content need to be first retrieved with the datalad get command.

Plotting the adjacency matrices

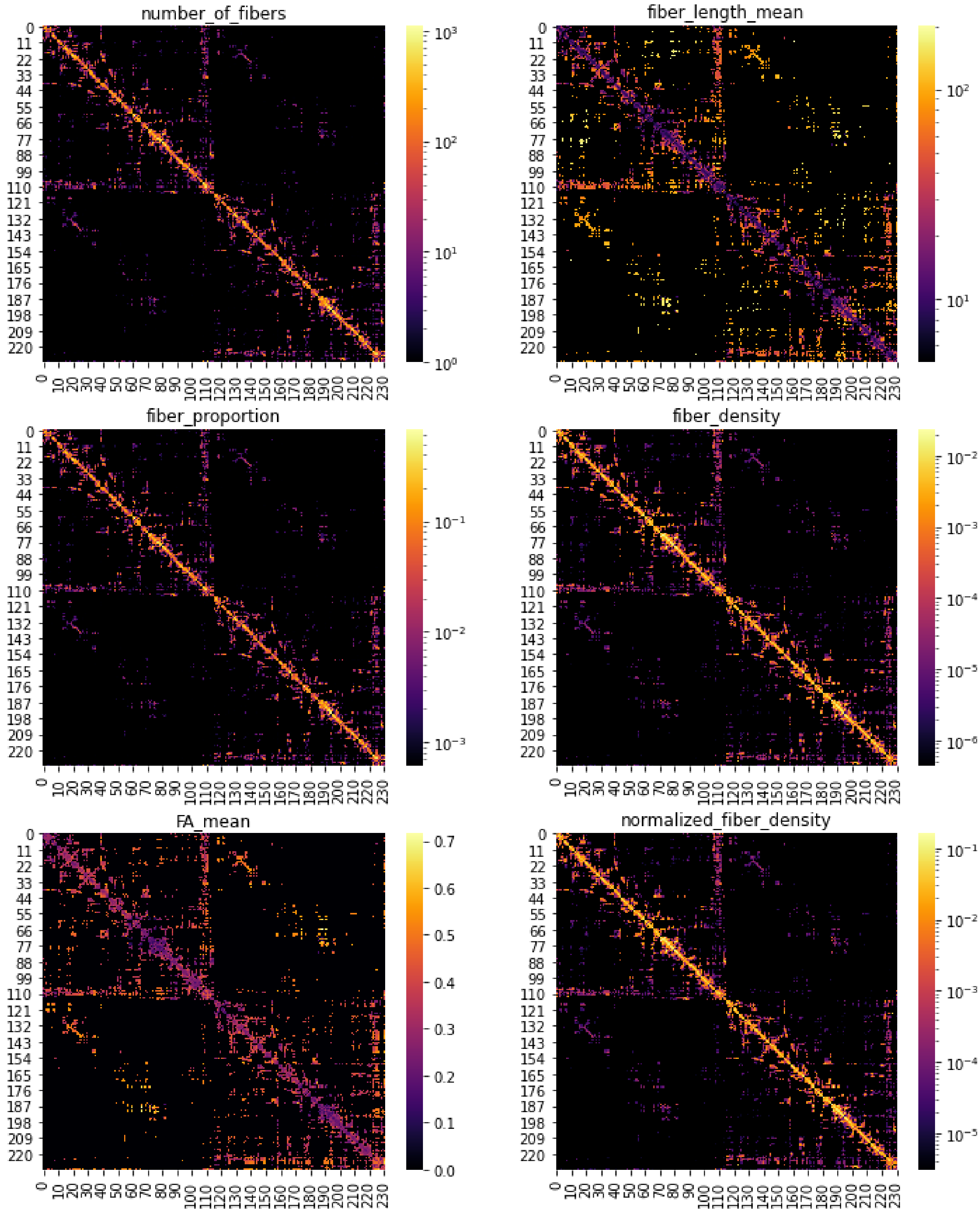
Let's plot some of those adjacency matrices using Matplotlib and Seaborn:

[10]:
```
# Create color map to handle zeros with log visualization
custom_cmap = copy.copy(plt.cm.get_cmap("inferno"))
# Copy the default cmap (0,0,0.5156)
custom_cmap.set_bad((0, 0, 0))

# Define the metrics to plot
cols_to_plot = ["number_of_fibers", "fiber_length_mean",
                "fiber_proportion", "fiber_density",
                "FA_mean", "normalized_fiber_density"]

# Plot with log-scale for all metrics except FA_mean
fig, axs = plt.subplots(3,2, figsize=(12,15))
axs = axs.flatten()
for c, ax in zip(cols_to_plot, axs):
    A = nx.to_numpy_array(G, weight=c)
    sns.heatmap(A, ax=ax, cmap=custom_cmap, norm=(LogNorm()
                                                  if c != "FA_mean"
```

(continues on next page)

(continued from previous page)

```
                                                      else None))
    ax.set_title(c)
```



Graph signal processing with structural connectivity and visualization

The package PyGSP offers a range of graph signal processing tools we can use on our structural connectivity data. In particular, we can do an eigendecomposition of the graph Laplacian to get the Fourier basis - the connectome harmonics.

Even though it is possible to also do this for subcortical regions, for the sake of plotting it is easier just to work with the cortical regions. To identify those, we need the parcellation labels.

[11]:
```
# Query the BIDS index/label mapping TSV file for the corresponding scale
label_path = layout.get(subject='01',
                    datatype='anat',
                    atlas='L2018',
                    res='scale3',
                    suffix='dseg',
                    extension='tsv',
                    return_type='filename')
print(f' BIDS index/label mapping TSV filepath: {label_path[0]}')
```

```
 BIDS index/label mapping TSV filepath: /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.0.3/sub-01/anat/sub-
↪01_atlas-L2018_res-scale3_dseg.tsv
```

[12]:
```
# Load the TSV content
labels = pd.read_csv(label_path[0], sep="\t", index_col=0)
# Reset index to start at 0
labels.reset_index(inplace=True)
# Select cortex labels
labels_ctx = labels["name"][[n.startswith("ctx") for n in labels["name"]]].copy()
idx = list(labels_ctx.index)
# Select rows with cortical areas
# A_fd_ctx = A_fiber_density[idx]
A = nx.to_numpy_array(G, weight="FA_mean")
A_fd_ctx = A[idx]
# Select columns with cortical areas
A_fd_ctx = A_fd_ctx[:,idx]
```

[13]:
```
# Display the shape of the matrix
A_fd_ctx.shape
```

[13]:
```
(216, 216)
```

Now we can compute the harmonics:

[14]:
```
np.fill_diagonal(A_fd_ctx, 0)  # PyGSP does not support self-loops
G_fd = pygsp.graphs.Graph(A_fd_ctx)  # PyGSP graph
G_fd.compute_laplacian(lap_type="normalized")
G_fd.compute_fourier_basis()  # compute connectome harmonics
```

The harmonics have the same dimensions as our original adjacency matrix.

[15]:
```
# Display the shape of the matrix
G_fd.U.shape
```

[15]:
```
(216, 216)
```

Each column contains one basis vector.

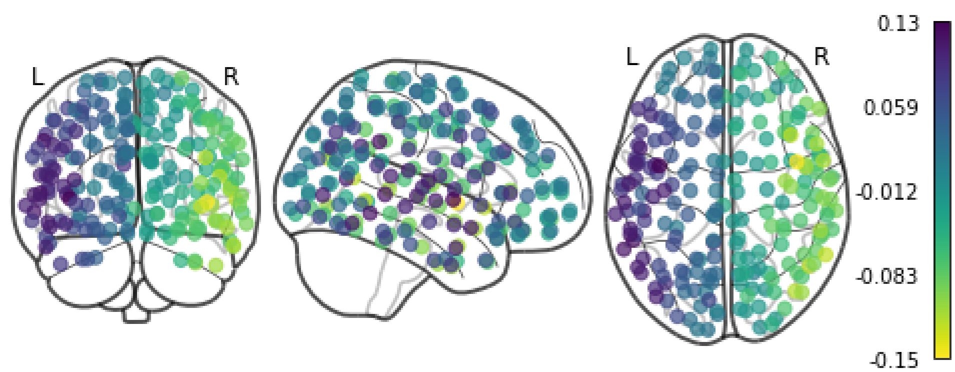
Basic visualization with Nilearn

Nilearn offers a quick and easy way to plot them using `plot_markers`. For this, we need the center coordinates of each region in the parcellation in MNI space. For your convenience, they have been already computed and can be easily retrieved with the `get_lausanne2018_parcellation_mni_coords(scale)` utility function of CMP3.

[16]:
```
# Import the util function
from cmtklib.data.parcellation.util import get_lausanne2018_parcellation_mni_coords
```

[17]:
```
# Load coordinates with the utility function provided by CMP3
coords_ctx = get_lausanne2018_parcellation_mni_coords('scale3')
# Plot
plotting.plot_markers(G_fd.U[:,1], coords_ctx)
```

[17]:
```
<nilearn.plotting.displays.OrthoProjector at 0x7f847a3e5690>
```



Advanced visualization with Nilearn

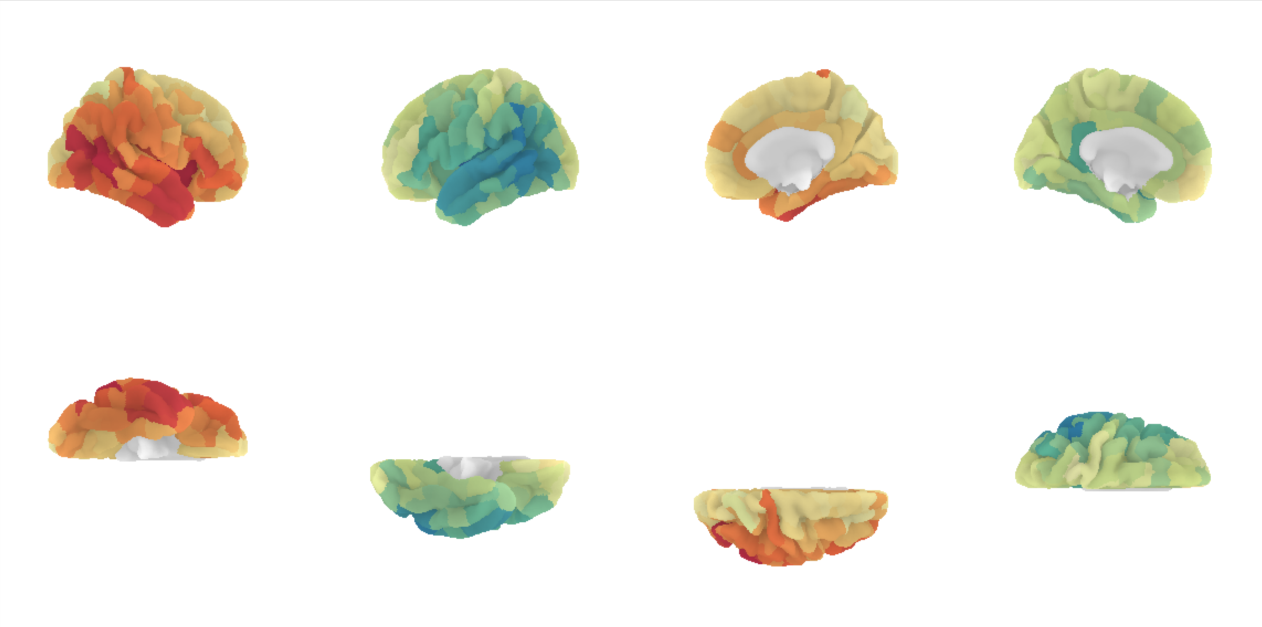
A prettier version is to plot the connectome harmonics on a brain surface using Nilearn `plot_surf_roi()`. For your convenience, a multiple view plot can be easily generated and saved with the `plot_lausanne2018_surface_ctx()` of the `cmtklib.data.parcellation.viz` module of CMP3, by specifying the scale to use.

These figures take a few minutes to generate, so you might need to be a bit patient.

[18]:
```
# Import the viz function
from cmtklib.data.parcellation.viz import plot_lausanne2018_surface_ctx
```

[19]:
```
%%time
# Plot
plot_lausanne2018_surface_ctx(G_fd.U[:,1], scale='scale3', save_fig=True)
```

```
CPU times: user 1min 37s, sys: 5.73 s, total: 1min 43s
Wall time: 1min 23s
```



Pretty, right? This concludes the tutorial. We hope you enjoy it and any feedback or suggestions to improve it are very welcome! Just please open a new issue on GitHub and share your thoughts with us.

Want to learn more about connectome harmonics?

Here are some references:

- Human brain networks function in connectome-specific harmonic waves (Atasoy et al., 2016, link): Landmark paper that first applied graph signal processing to brain connectivity.
- Functional harmonics reveal multi-dimensional basis functions underlying cortical organization (Glomb et al., 2021, link): Connectome harmonics of functional connectivity.
- The connectome spectrum as a canonical basis for a sparse representation of fast brain activity (Rué-Queralt et al., 2021, link): EEG and connectome harmonics.

```
[ ]:
```

5.9.2 EEG Pipeline Tutorial

In this notebook, we will demonstrate how to use the newly implemented EEG pipeline of CMP3, using the "VEPCON" dataset, available at https://openneuro.org/datasets/ds003505/versions/1.1.1.

It is important to note that CMP3 *does not* include preprocessing of EEG data, so it is expected that you have your data ready to be analyzed.

Important: Note that the skull-surfaces provided with the dataset ("bem", see below) which are needed to create the head model are obtained from non-defaced MRIs. You will not be able to proceed with surfaces created from VEPCON dataset alone.

Setup instructions

If you want to reproduce all the results of this notebook on your side, a conda `environment.yml` file can be downloaded at the following link: EEG_tutorial_environment.yml. The original `.ipynb` notebook file can be downloaded at the following link: EEG_pipeline_tutorial.ipynb.

Once you have downloaded the conda environment file, install the environment `py37cmp-eeg` as follows:

```
$ conda env create -f /path/to/downloaded/EEG_tutorial_environment.yml
```

This will install all the packages needed to run this notebook including jupyter lab.

You can then activate it, go to the directory where you downloaded the `EEG_pipeline_tutorial.ipynb`, and launch jupyter lab as follows:

```
$ cd /directory/of/downloaded/EEG_pipeline_tutorial.ipynb
$ conda activate py37cmp-eeg
$ jupyter lab
```

You are ready to open and interact with the notebook!

[ ]:
```
!pip install ../../.
```

Loading the python packages used in the notebook

[1]:
```
# General
import sys
import os
import argparse
import subprocess
import pdb
import pickle
import shutil
import json
from IPython.display import SVG, display
import warnings

# Dataset management
import datalad.api as dl

# Data/graph handling and visualization
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx

# BIDS import
from bids import BIDSLayout

# MNE imports
import mne
import mne_connectivity as mnec

# CMP3 imports
```

(continues on next page)

(continued from previous page)

```
import cmp.project
from cmp.info import __version__, __copyright__
from cmtklib.util import print_error, print_blue, print_warning
from cmtklib.bids.io import (
    __nipype_directory__, __cartool_directory__,
    __eeglab_directory__, __cmp_directory__
)
# other
from EEG_tutorial_utils import (
    create_trans_files,
    fix_vepcon_derivatives_dataset_description_files
)
```

```
220709-17:01:15,509 nipype.utils WARNING:
         A newer version (1.8.1) of nipy/nipype is available. You are using 1.7.0
```

Loading the BIDS dataset

For demonstration, we are going to use the latest version of VEPCON dataset, available on Open Neuro that already contains outputs from Connectome Mapper v3.0.3 and Freesurfer 7.1.1. A full description of the dataset can be found in Pascucci, Tourbier, et al. 2022.

In case you want to rerun the notebook, make sure to remove any `ds003505_demo` folder in the directory of the notebook. Otherwise, datalad install will complain.

[2]:
```
%%time
# Download example dataset with datalad
bids_dir = os.path.join(".", "ds003505_demo")  # Adjust path to your BIDS dataset as␣
↪needed
vepcon_data = dl.install(
    path=bids_dir,
    source="https://github.com/OpenNeuroDatasets/ds003505.git"
)
```

```
CPU times: user 9.95 ms, sys: 13.8 ms, total: 23.7 ms
Wall time: 58.6 ms
```

Running the EEG pipeline

As of now, the EEG pipeline can only be run directly from the application programming interface (API) as demonstrated in this notebook. As soon as possible, we will integrate it into the graphical user interface (GUI) and the command line interface (CLI).

First, we need to configure the following user-defined arguments. Please modify them as needed.

[3]:
```
# Adjust path to your BIDS dataset as needed
bids_dir = vepcon_data.path

# Adjust path of the output directory as needed
output_dir = os.path.join(bids_dir, 'derivatives')

# Adjust the subject to be processed as needed
```

(continues on next page)

(continued from previous page)

```
participant_label = 'sub-01'

# Adjust the name of the task to be considered
task_label = 'faces'

# Adjust path to the anatomical pipeline configuration file as needed
anat_pipeline_config = os.path.join(bids_dir, 'code', 'ConnectomeMapper-Docker', 'ref_
↪anatomical_config.json')

# Adjust path to the MNE-based pipeline configuration file as needed
eeg_pipeline_config =  os.path.join('.', 'ref_mne_eeg_config.json')
```

The eeg pipeline config .json file contains information that CMP3 needs to correctly load EEG data and associated information like electrode positions, names of conditions, which parcellation to use, etc. as seen below:

[4]:
```
%cat ref_mne_eeg_config.json
```


```
{
    "Global": {
        "process_type": "EEG",
        "subjects": [
            "sub-01"
        ],
        "subject": "sub-01",
        "version": "v3.1.0"
    },
    "eeg_preprocessing_stage": {
        "task_label": "faces",
        "eeg_ts_file.extension": "set",
        "eeg_ts_file.toolbox_derivatives_dir": "eeglab-v14.1.1",
        "eeg_ts_file.datatype": "eeg",
        "eeg_ts_file.suffix": "eeg",
        "eeg_ts_file.desc": "preproc",
        "eeg_ts_file.task": "faces",
        "events_file.datatype": "eeg",
        "events_file.suffix": "events",
        "events_file.extension": "tsv",
        "events_file.task": "faces",
        "electrodes_file_fmt": "Cartool",
        "cartool_electrodes_file.toolbox_derivatives_dir": "cartool-v3.80",
        "cartool_electrodes_file.datatype": "eeg",
        "cartool_electrodes_file.suffix": "eeg",
        "cartool_electrodes_file.extension": "xyz",
        "t_min": -0.2,
        "t_max": 0.5
    },
    "eeg_source_imaging_stage": {
        "esi_tool": "MNE",
        "mne_apply_electrode_transform": true,
        "mne_electrode_transform_file.toolbox_derivatives_dir": "cmp-v3.0.3",
        "mne_electrode_transform_file.datatype": "eeg",
        "mne_electrode_transform_file.suffix": "trans",
        "mne_electrode_transform_file.extension": "fif",
```


(continues on next page)

(continued from previous page)

```
        "parcellation_cmp_dir": "cmp-v3.0.3",
        "parcellation_scheme": "Lausanne2018",
        "lausanne2018_parcellation_res": "scale1",
        "mne_esi_method": "sLORETA",
        "mne_esi_method_snr": 3.0
    },
    "eeg_connectome_stage": {
        "connectivity_metrics": [
            "coh",
            "cohy",
            "imcoh",
            "plv",
            "ciplv",
            "ppc",
            "pli",
            "wpli",
            "wpli2_debiased"
        ],
        "output_types": [
            "tsv",
            "gpickle",
            "mat",
            "graphml"
        ]
    },
    "Multi-processing": {
        "number_of_cores": 1
    }
}
```

Note: If you would like to run another subject (all available subjects except subjects 5 and 15 can be run), you will need to modify the config files (replacing `sub-<label>` accordingly).

Then, we need to tell datalad to download the actual content of the structural MRI and EEG files that will be input to the pipelines.

[5]:
```
%%time
# Raw MRI
vepcon_data.get(f'{participant_label}/anat/')
```

```
CPU times: user 7.09 ms, sys: 9.96 ms, total: 17 ms
Wall time: 135 ms
```

[5]:
```
[{'action': 'get',
  'path': '/Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/sub-01/anat',
  'type': 'directory',
  'refds': '/Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo',
  'status': 'notneeded',
  'message': ('nothing to get from %s',
   '/Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/sub-01/anat')}]
```

(continues on next page)

(continued from previous page)

[6]:
```
%%time
# CMP3 and Freesurfer derivatives
vepcon_data.get(f'derivatives/cmp-v3.0.3/{participant_label}/anat/')
vepcon_data.get(f'derivatives/freesurfer-7.1.1/{participant_label}/')
```

```
CPU times: user 7.82 ms, sys: 10.2 ms, total: 18.1 ms
Wall time: 165 ms
```

[6]:
```
[{'action': 'get',
  'path': '/Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/freesurfer-7.1.1/sub-01',
  'type': 'directory',
  'refds': '/Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo',
  'status': 'notneeded',
  'message': ('nothing to get from %s',
   '/Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/freesurfer-7.1.1/sub-01')}]
```

[7]:
```
%%time
# Electrode position
vepcon_data.get(f'derivatives/{__cartool_directory__}/{participant_label}/' +
                f'eeg/{participant_label}_eeg.xyz')
# Preprocessed EEG in EEGLab .fdt/.set format
vepcon_data.get(f'derivatives/{__eeglab_directory__}/{participant_label}/eeg/' +
                f'{participant_label}_task-faces_desc-preproc_eeg.fdt')
vepcon_data.get(f'derivatives/{__eeglab_directory__}/{participant_label}/eeg/' +
                f'{participant_label}_task-faces_desc-preproc_eeg.set')
```

```
CPU times: user 11.1 ms, sys: 14.5 ms, total: 25.6 ms
Wall time: 223 ms
```

[7]:
```
[{'action': 'get',
  'path': '/Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/eeglab-v14.1.1/sub-01/eeg/sub-01_task-faces_desc-preproc_eeg.
↪set',
  'type': 'file',
  'refds': '/Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo',
  'status': 'notneeded',
  'message': 'already present'}]
```

In the latest version of VEPCON (`v1.1.1`) that we use in this tutorial, we notice that `dataset_description.json` files in the `derivatives/cartool-v3.80` and `derivatives/eeglab-v14.1.1` are invalid and will create an error if these directories are added to the BIDSLayout representation of the VEPCON dataset. We need to fix them by running the following helper function provided along this tutorial:

[8]:
```
fix_vepcon_derivatives_dataset_description_files(os.path.abspath(bids_dir))
```

```
Replace /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cartool-v3.80/dataset_description.json
Replace /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/eeglab-v14.1.1/dataset_description.json
```

We can then configure a new CMP3 project.

[9]:
```
# initialize project
project = cmp.project.ProjectInfo()
project.base_directory = os.path.abspath(bids_dir)
project.output_directory = os.path.abspath(output_dir)
project.subjects = ["{}".format(participant_label)]
project.subject = "{}".format(participant_label)

# VEPCON dataset does not have a subject/sessions structure
project.subject_sessions = [""]
project.subject_session = ""

# Set the path to the anatomical pipeline configuration file
project.anat_config_file = os.path.abspath(anat_pipeline_config)
```

As the dataset is in BIDS, we can use Pybids to help us with the task of interacting with the files of the dataset.

[10]:
```
# Represent the BIDS dataset as a PyBIDS BIDSLayout
bids_layout = BIDSLayout(project.base_directory)
```

Once set, we can run the anatomical pipeline, in order to obtain, among other things, Freesurfer derivatives necessary for the MNE pipeline.

Freesurfer and CMP3 derivatives are indeed provided with the VEPCON dataset, so we do not need to run it, but if run on a fresh dataset

[11]:
```
%%time
# Do not run again the anatomical pipeline
# You will have to set it to True on a fresh dataset
run = False

# Initialize the anatomical pipeline reading the configuration file
anat_pipeline = cmp.project.init_anat_project(project, False)

if anat_pipeline is not None:
    # Check if inputs to anatomical pipeline are valid
    anat_valid_inputs = anat_pipeline.check_input(bids_layout, gui=False)
    if anat_valid_inputs:
        if run:
            print(">> Process anatomical pipeline")
            anat_pipeline.process()
    else:
        print_error("  .. ERROR: Invalid inputs")
        exit_code = 1

# Check if outputs to anatomical pipeline are valid
if run:
    anat_valid_outputs, msg = anat_pipeline.check_output()
else:
    anat_valid_outputs = True

# Set the freesurfer subjects directory and the subject id
project.freesurfer_subjects_dir = anat_pipeline.stages['Segmentation'].config.freesurfer_
↪subjects_dir
```

(continues on next page)

(continued from previous page)

```
project.freesurfer_subject_id = anat_pipeline.stages['Segmentation'].config.freesurfer_
↪subject_id
```

```
  .. LOAD: Load anatomical config file : /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/code/ConnectomeMapper-Docker/ref_
↪anatomical_config.json
  .. WARNING: CMP3 version used to generate the configuration files (v3.0.2)  and␣
↪version of CMP3 used (v3.1.0) differ
**** Check Inputs  ****
> Looking in /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo for...
/Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/
↪sub-01/anat/sub-01_T1w.nii.gz
... t1_file : /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/sub-01/anat/sub-01_T1w.nii.gz
/Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/
↪sub-01/anat/sub-01_T1w.json
... t1_json_file : /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
↪notebooks/ds003505_demo/sub-01/anat/sub-01_T1w.json
Inputs check finished successfully.
Only anatomical data (T1) available.
CPU times: user 29.6 ms, sys: 6.12 ms, total: 35.7 ms
Wall time: 33.1 ms
```

In VEPCON, the electrode positions are provided in a file in the Cartool-derivatives folder, but CMP3 expects them in the EEGLAB-derivatives folder.

[12]:
```
# To be Removed !!!
# Copy the file to the appropriate location
#cartool_file_location = os.path.join(
#    bids_dir, 'derivatives', __cartool_directory__,
#    participant_label,'eeg', participant_label + '_eeg.xyz'
#)
#eeglab_file_location = os.path.join(
#    bids_dir, 'derivatives', 'eeglab-v14.1.1',
#    participant_label, 'eeg', participant_label + '_eeg.xyz')

# if not os.path.exists(eeglab_file_location):
#    _ = shutil.copyfile(cartool_file_location, eeglab_file_location)
```

Since we are using non-defaced MRIs, which are not exactly the same as the ones provided on OpenNeuro, we need an additional transform that will be applied to the electrode positions.

[13]:
```
# The following line creates the appropriate file with this transform in derivatives/cmp-
↪v3.0.3:
create_trans_files(bids_dir, participant_label)
```

```
Overwriting existing file.
```

Finally, you can run the EEG pipeline.

[14]:
```
%%time

from cmtklib import config
```

(continues on next page)

(continued from previous page)

```
# IF on MacOSX, add /usr/sbin to the $PATH
# which contains sysctl
# Otherwise, Nipype raises an "/bin/sh: sysctl: command not found" error
# when trying to get the system memory
if "darwin" in sys.platform:
    os.environ["PATH"] = f'/usr/sbin/:{os.environ["PATH"]}'
# Note that "sysctl" can be located in a different place
# than "/usr/sbin".
# To know which path has to be added, you can run
# `locate sysctl`

# Set the path to the anatomical pipeline configurration file
eeg_pipeline_config = 'ref_mne_eeg_config.json'
project.eeg_config_file = os.path.abspath(eeg_pipeline_config)

if anat_valid_outputs:
    # Initialize the EEG pipeline reading the configuration file and
    # check input validity
    eeg_valid_inputs, eeg_pipeline = cmp.project.init_eeg_project(
        project, False
    )
    if eeg_pipeline is not None:
        eeg_pipeline.parcellation_scheme = anat_pipeline.parcellation_scheme
        eeg_pipeline.atlas_info = anat_pipeline.atlas_info
        eeg_pipeline.stages['EEGPreprocessing'].config.task_label = 'faces'

        if eeg_valid_inputs:
            print(">> Process EEG pipeline")
            eeg_pipeline.process()
        else:
            print("  .. ERROR: Invalid inputs")
            exit_code = 1
else:
    print_error(f'  .. ERROR: Invalid anatomical outputs for eeg pipeline')
    print_error(f'{msg}')
    exit_code = 1
```

```
**** Check Inputs ****
Base dir: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline
 .. DEBUG : Generated file name = sub-01_atlas-L2018_res-scale1_dseg.nii.gz
 .. DEBUG : Generated file name = sub-01_atlas-L2018_res-scale1_dseg.nii.gz
cmp-v3.0.3
220709-17:01:27,860 nipype.workflow INFO:
         [Node] Setting-up "eeg_check_input" in "/Users/sebastientourbier/Documents/
↪GitHub/connectomemapper3/docs/notebooks/ds003505_demo/derivatives/nipype-1.7.0/sub-01/
↪eeg_pipeline/eeg_check_input".
220709-17:01:27,869 nipype.workflow INFO:
         [Node] Executing "eeg_check_input" <nipype.interfaces.io.BIDSDataGrabber>
Load dataset_description for: /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.0.3
```

(continues on next page)

(continued from previous page)

```
Load dataset_description for: /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/eeglab-v14.1.1
Load dataset_description for: /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cartool-v3.80
220709-17:01:36,614 nipype.workflow INFO:
	 [Node] Finished "eeg_check_input", elapsed time 8.742065s.
	.. Input file for "eeg_ts_file" key: ['/Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/eeglab-v14.1.1/sub-01/eeg/
↪sub-01_task-faces_desc-preproc_eeg.set']
	.. Input file for "events_file" key: ['/Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/sub-01/eeg/sub-01_task-faces_events.tsv
↪']
	.. Input file for "electrodes_file" key: ['/Users/sebastientourbier/Documents/
↪GitHub/connectomemapper3/docs/notebooks/ds003505_demo/sub-01/eeg/sub-01_task-faces_
↪electrodes.tsv']
	.. Input file for "roi_volume_file" key: ['/Users/sebastientourbier/Documents/
↪GitHub/connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.0.3/sub-01/
↪anat/sub-01_atlas-L2018_res-scale1_dseg.nii.gz', '/Users/sebastientourbier/Documents/
↪GitHub/connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.0.3/sub-01/
↪anat/sub-01_space-DWI_atlas-L2018_res-scale1_dseg.nii.gz']
  .. LOAD: Load EEG config file : /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ref_mne_eeg_config.json
  .. INFO: Generated with the same CMP3 version
{'Global': {'process_type': 'EEG', 'subjects': ['sub-01'], 'subject': 'sub-01', 'version
↪': 'v3.1.0'}, 'eeg_preprocessing_stage': {'task_label': 'faces', 'eeg_ts_file.extension
↪': 'set', 'eeg_ts_file.toolbox_derivatives_dir': 'eeglab-v14.1.1', 'eeg_ts_file.
↪datatype': 'eeg', 'eeg_ts_file.suffix': 'eeg', 'eeg_ts_file.desc': 'preproc', 'eeg_ts_
↪file.task': 'faces', 'events_file.datatype': 'eeg', 'events_file.suffix': 'events',
↪'events_file.extension': 'tsv', 'events_file.task': 'faces', 'electrodes_file_fmt':
↪'Cartool', 'cartool_electrodes_file.toolbox_derivatives_dir': 'cartool-v3.80',
↪'cartool_electrodes_file.datatype': 'eeg', 'cartool_electrodes_file.suffix': 'eeg',
↪'cartool_electrodes_file.extension': 'xyz', 't_min': -0.2, 't_max': 0.5}, 'eeg_source_
↪imaging_stage': {'esi_tool': 'MNE', 'mne_apply_electrode_transform': True, 'mne_
↪electrode_transform_file.toolbox_derivatives_dir': 'cmp-v3.0.3', 'mne_electrode_
↪transform_file.datatype': 'eeg', 'mne_electrode_transform_file.suffix': 'trans', 'mne_
↪electrode_transform_file.extension': 'fif', 'parcellation_cmp_dir': 'cmp-v3.0.3',
↪'parcellation_scheme': 'Lausanne2018', 'lausanne2018_parcellation_res': 'scale1', 'mne_
↪esi_method': 'sLORETA', 'mne_esi_method_snr': 3.0}, 'eeg_connectome_stage': {
↪'connectivity_metrics': ['coh', 'cohy', 'imcoh', 'plv', 'ciplv', 'ppc', 'pli', 'wpli',
↪'wpli2_debiased'], 'output_types': ['tsv', 'gpickle', 'mat', 'graphml']}, 'Multi-
↪processing': {'number_of_cores': 1}}
>> Process EEG pipeline
220709-17:01:36,634 nipype.interface INFO:
	 **** Processing ****
 .. DEBUG : Generated file name = sub-01_atlas-L2018_res-scale1_dseg.nii.gz
 .. DEBUG : Generated file path (no extension) = /Users/sebastientourbier/Documents/
↪GitHub/connectomemapper3/docs/notebooks/ds003505_demo/sub-01/eeg/sub-01_task-faces_
↪events
  .. DEBUG: Event_ids for Epochs extraction: {'SCRAMBLED': '0', 'FACES': '1'}
220709-17:01:37,164 nipype.workflow INFO:
	 Generated workflow graph: /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_
↪pipeline/graph.svg (graph2use=colored, simple_form=True).
```

(continues on next page)

(continued from previous page)

```
220709-17:01:37,205 nipype.workflow INFO:
	 Workflow eeg_pipeline settings: ['check', 'execution', 'logging', 'monitoring']
220709-17:01:37,219 nipype.workflow INFO:
	 Running in parallel.
220709-17:01:37,222 nipype.workflow INFO:
	 [MultiProc] Running 0 tasks, and 3 jobs ready. Free memory (GB): 14.40/14.40,␣
↪Free processors: 1/1.
220709-17:01:37,344 nipype.workflow INFO:
	 [Node] Outdated cache found for "eeg_pipeline.eeg_datasource".
220709-17:01:37,365 nipype.workflow INFO:
	 [Node] Setting-up "eeg_pipeline.eeg_datasource" in "/Users/sebastientourbier/
↪Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/derivatives/nipype-1.7.
↪0/sub-01/eeg_pipeline/eeg_datasource".
220709-17:01:37,371 nipype.workflow INFO:
	 [Node] Outdated cache found for "eeg_pipeline.eeg_datasource".
220709-17:01:37,380 nipype.workflow INFO:
	 [Node] Executing "eeg_datasource" <nipype.interfaces.io.BIDSDataGrabber>
220709-17:01:39,229 nipype.workflow INFO:
	 [MultiProc] Running 1 tasks, and 2 jobs ready. Free memory (GB): 14.20/14.40,␣
↪Free processors: 0/1.
                     Currently running:
                       * eeg_pipeline.eeg_datasource
Load dataset_description for: /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/eeglab-v14.1.1
Load dataset_description for: /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cartool-v3.80
Load dataset_description for: /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.0.3
220709-17:01:46,693 nipype.workflow INFO:
	 [Node] Finished "eeg_datasource", elapsed time 9.308715s.
220709-17:01:47,241 nipype.workflow INFO:
	 [Job 0] Completed (eeg_pipeline.eeg_datasource).
220709-17:01:47,249 nipype.workflow INFO:
	 [MultiProc] Running 0 tasks, and 3 jobs ready. Free memory (GB): 14.40/14.40,␣
↪Free processors: 1/1.
220709-17:01:47,431 nipype.workflow INFO:
	 [Node] Setting-up "eeg_pipeline.eeg_source_imaging_stage.mne_createbem" in "/
↪Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_source_imaging_stage/mne_
↪createbem".
220709-17:01:47,437 nipype.workflow INFO:
	 [Node] Executing "mne_createbem" <cmtklib.interfaces.mne.CreateBEM>
Creating the BEM geometry...
Going from 5th to 4th subdivision of an icosahedron (n_tri: 20480 -> 5120)
Going from 5th to 4th subdivision of an icosahedron (n_tri: 20480 -> 5120)
Going from 5th to 4th subdivision of an icosahedron (n_tri: 20480 -> 5120)
outer skin  CM is  -1.05  -8.88  11.57 mm
outer skull CM is  -1.05  -8.80  11.09 mm
inner skull CM is  -1.05 -10.32  19.78 mm
Checking that surface outer skull is inside surface outer skin  ...
220709-17:01:49,241 nipype.workflow INFO:
	 [MultiProc] Running 1 tasks, and 2 jobs ready. Free memory (GB): 14.20/14.40,␣
↪Free processors: 0/1.
```

(continues on next page)

(continued from previous page)

```
                     Currently running:
                       * eeg_pipeline.eeg_source_imaging_stage.mne_createbem
Checking that surface inner skull is inside surface outer skull ...
Checking distance between outer skin  and outer skull surfaces...
Minimum distance between the outer skin  and outer skull surfaces is approximately    1.
↪6 mm
Checking distance between outer skull and inner skull surfaces...
Minimum distance between the outer skull and inner skull surfaces is approximately    1.
↪8 mm
Surfaces passed the basic topology checks.
Complete.

Approximation method : Linear collocation

Three-layer model surfaces loaded.
Computing the linear collocation solution...
    Matrix coefficients...
        outer skin  (2562) -> outer skin  (2562) ...
        outer skin  (2562) -> outer skull (2562) ...
        outer skin  (2562) -> inner skull (2562) ...
        outer skull (2562) -> outer skin  (2562) ...
        outer skull (2562) -> outer skull (2562) ...
        outer skull (2562) -> inner skull (2562) ...
        inner skull (2562) -> outer skin  (2562) ...
        inner skull (2562) -> outer skull (2562) ...
        inner skull (2562) -> inner skull (2562) ...
    Inverting the coefficient matrix...
IP approach required...
    Matrix coefficients (homog)...
        inner skull (2562) -> inner skull (2562) ...
    Inverting the coefficient matrix (homog)...
    Modify the original solution to incorporate IP approach...
        Combining...
        Scaling...
Solution ready.
BEM geometry computations complete.
220709-17:02:50,820 nipype.workflow INFO:
         [Node] Finished "mne_createbem", elapsed time 63.377566s.
220709-17:02:51,340 nipype.workflow INFO:
         [Job 1] Completed (eeg_pipeline.eeg_source_imaging_stage.mne_createbem).
220709-17:02:51,346 nipype.workflow INFO:
         [MultiProc] Running 0 tasks, and 2 jobs ready. Free memory (GB): 14.40/14.40,␣
↪Free processors: 1/1.
220709-17:02:51,455 nipype.workflow INFO:
         [Node] Setting-up "eeg_pipeline.eeg_source_imaging_stage.mne_createsrc" in "/
↪Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_source_imaging_stage/mne_
↪createsrc".
220709-17:02:51,460 nipype.workflow INFO:
         [Node] Executing "mne_createsrc" <cmtklib.interfaces.mne.CreateSrc>
Setting up the source space with the following parameters:
```

(continues on next page)

(continued from previous page)

```
SUBJECTS_DIR = /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
↪notebooks/ds003505_demo/derivatives/freesurfer-7.1.1
Subject      = sub-01
Surface      = white
Octahedron subdivision grade 6

>>> 1. Creating the source space...

Doing the octahedral vertex picking...
Loading /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/freesurfer-7.1.1/sub-01/surf/lh.white...
Mapping lh sub-01 -> oct (6) ...
    Triangle neighbors and vertex normals...
Loading geometry from /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
↪notebooks/ds003505_demo/derivatives/freesurfer-7.1.1/sub-01/surf/lh.sphere...
Setting up the triangulation for the decimated surface...
loaded lh.white 4098/149863 selected to source space (oct = 6)

Loading /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/freesurfer-7.1.1/sub-01/surf/rh.white...
Mapping rh sub-01 -> oct (6) ...
    Triangle neighbors and vertex normals...
Loading geometry from /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
↪notebooks/ds003505_demo/derivatives/freesurfer-7.1.1/sub-01/surf/rh.sphere...
Setting up the triangulation for the decimated surface...
loaded rh.white 4098/147183 selected to source space (oct = 6)

Calculating source space distances (limit=inf mm)...
220709-17:02:53,344 nipype.workflow INFO:
         [MultiProc] Running 1 tasks, and 1 jobs ready. Free memory (GB): 14.20/14.40,␣
↪Free processors: 0/1.
                     Currently running:
                       * eeg_pipeline.eeg_source_imaging_stage.mne_createsrc
    Computing patch statistics...
    Patch information added...
    Computing patch statistics...
    Patch information added...
You are now one step closer to computing the gain matrix
    Write a source space...
    [done]
    Write a source space...
    [done]
    2 source spaces written
220709-17:13:09,333 nipype.workflow INFO:
         [Node] Finished "mne_createsrc", elapsed time 617.869614s.
220709-17:13:10,258 nipype.workflow INFO:
         [Job 2] Completed (eeg_pipeline.eeg_source_imaging_stage.mne_createsrc).
220709-17:13:10,264 nipype.workflow INFO:
         [MultiProc] Running 0 tasks, and 1 jobs ready. Free memory (GB): 14.40/14.40,␣
↪Free processors: 1/1.
220709-17:13:10,360 nipype.workflow INFO:
         [Node] Setting-up "eeg_pipeline.eeg_preprocessing_stage.eeglab2fif" in "/Users/
↪sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/
↪derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_preprocessing_stage/eeglab2fif".
```

(continues on next page)

(continued from previous page)

```
220709-17:13:10,367 nipype.workflow INFO:
	 [Node] Executing "eeglab2fif" <cmtklib.interfaces.mne.EEGLAB2fif>

eeg_ts_file = /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/eeglab-v14.1.1/sub-01/eeg/sub-01_task-faces_desc-preproc_eeg.
↪set
electrodes_file = /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
↪notebooks/ds003505_demo/derivatives/cartool-v3.80/sub-01/eeg/sub-01_eeg.xyz
event_ids = {'SCRAMBLED': '0', 'FACES': '1'}
events_file = /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/sub-01/eeg/sub-01_task-faces_events.tsv
out_epochs_fif_fname = epo.fif
t_max = 0.5
t_min = -0.2

Extracting parameters from /Users/sebastientourbier/Documents/GitHub/connectomemapper3/
↪docs/notebooks/ds003505_demo/derivatives/eeglab-v14.1.1/sub-01/eeg/sub-01_task-faces_
↪desc-preproc_eeg.set...
Not setting metadata
Not setting metadata
588 matching events found
No baseline correction applied
0 projection items activated
Ready.
Applying baseline correction (mode: mean)
Adding average EEG reference projection.
1 projection items deactivated
Average reference projection was added, but has not been applied yet. Use the apply_proj␣
↪method to apply it.
.. INFO: montage_fname = /Users/sebastientourbier/Documents/GitHub/connectomemapper3/
↪docs/notebooks/ds003505_demo/derivatives/cartool-v3.80/sub-01/eeg/sub-01_eeg.xyz
	.. INFO: Create montage from Cartool electrodes file...
220709-17:13:11,693 nipype.workflow INFO:
	 [Node] Finished "eeglab2fif", elapsed time 1.322598s.
220709-17:13:12,258 nipype.workflow INFO:
	 [Job 3] Completed (eeg_pipeline.eeg_preprocessing_stage.eeglab2fif).
220709-17:13:12,264 nipype.workflow INFO:
	 [MultiProc] Running 0 tasks, and 2 jobs ready. Free memory (GB): 14.40/14.40,␣
↪Free processors: 1/1.
220709-17:13:12,361 nipype.workflow INFO:
	 [Node] Setting-up "eeg_pipeline.eeg_source_imaging_stage.mne_createcov" in "/
↪Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_source_imaging_stage/mne_
↪createcov".
220709-17:13:12,366 nipype.workflow INFO:
	 [Node] Executing "mne_createcov" <cmtklib.interfaces.mne.CreateCov>
Reading /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_preprocessing_stage/
↪eeglab2fif/epo.fif ...
    Read a total of 1 projection items:
        Average EEG reference (1 x 128)  idle
    Found the data of interest:
```

(continues on next page)

(continued from previous page)

```
        t =    -200.00 ...     500.00 ms
        0 CTF compensation matrices available
Not setting metadata
Not setting metadata
588 matching events found
No baseline correction applied
Created an SSP operator (subspace dimension = 1)
1 projection items activated
Computing rank from data with rank=None
    Using tolerance 3.2e-11 (2.2e-16 eps * 128 dim * 1.1e+03  max singular value)
    Estimated rank (eeg): 127
    EEG: rank 127 computed from 128 data channels with 1 projector
    Created an SSP operator (subspace dimension = 1)
    Setting small EEG eigenvalues to zero (without PCA)
Reducing data rank from 128 -> 127
Estimating covariance using SHRUNK
220709-17:13:14,260 nipype.workflow INFO:
         [MultiProc] Running 1 tasks, and 1 jobs ready. Free memory (GB): 14.20/14.40,␣
↪Free processors: 0/1.
                     Currently running:
                       * eeg_pipeline.eeg_source_imaging_stage.mne_createcov
Done.
Estimating covariance using EMPIRICAL
Done.
Using cross-validation to select the best estimator.
Number of samples used : 29988
log-likelihood on unseen data (descending order):
   shrunk: 71.873
   empirical: -373.964
selecting best estimator: shrunk
[done]
220709-17:13:15,592 nipype.workflow INFO:
         [Node] Finished "mne_createcov", elapsed time 3.22245s.
220709-17:13:16,262 nipype.workflow INFO:
         [Job 4] Completed (eeg_pipeline.eeg_source_imaging_stage.mne_createcov).
220709-17:13:16,270 nipype.workflow INFO:
         [MultiProc] Running 0 tasks, and 1 jobs ready. Free memory (GB): 14.40/14.40,␣
↪Free processors: 1/1.
220709-17:13:16,364 nipype.workflow INFO:
         [Node] Setting-up "eeg_pipeline.eeg_source_imaging_stage.mne_createfwd" in "/
↪Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_source_imaging_stage/mne_
↪createfwd".
220709-17:13:16,370 nipype.workflow INFO:
         [Node] Executing "mne_createfwd" <cmtklib.interfaces.mne.CreateFwd>
Reading /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_preprocessing_stage/
↪eeglab2fif/epo.fif ...
    Read a total of 1 projection items:
        Average EEG reference (1 x 128)  idle
    Found the data of interest:
        t =    -200.00 ...     500.00 ms
```

(continues on next page)

(continued from previous page)

```
        0 CTF compensation matrices available
Not setting metadata
Not setting metadata
588 matching events found
No baseline correction applied
Created an SSP operator (subspace dimension = 1)
1 projection items activated
Source space          : /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
↪notebooks/ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_source_
↪imaging_stage/mne_createsrc/src.fif
MRI -> head transform : /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
↪notebooks/ds003505_demo/derivatives/cmp-v3.0.3/sub-01/eeg/sub-01_trans.fif
Measurement data      : instance of Info
Conductor model   : /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
↪notebooks/ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_source_
↪imaging_stage/mne_createbem/bem.fif
Accurate field computations
Do computations in head coordinates
Free source orientations

Reading /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_source_imaging_stage/
↪mne_createsrc/src.fif...
Read 2 source spaces a total of 8196 active source locations

Coordinate transformation: MRI (surface RAS) -> head
     1.000000  0.000000  0.000000       0.00 mm
     0.000000  1.000000  0.000000       9.00 mm
     0.000000  0.000000  1.000000     -11.00 mm
     0.000000  0.000000  0.000000       1.00

Read 128 EEG channels from info
Head coordinate coil definitions created.
Source spaces are now in head coordinates.

Setting up the BEM model using /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_
↪pipeline/eeg_source_imaging_stage/mne_createbem/bem.fif...

Loading surfaces...

Loading the solution matrix...

Three-layer model surfaces loaded.
Loaded linear_collocation BEM solution from /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_
↪pipeline/eeg_source_imaging_stage/mne_createbem/bem.fif
Employing the head->MRI coordinate transform with the BEM model.
BEM model bem.fif is now set up

Source spaces are in head coordinates.
Checking that the sources are inside the surface (will take a few...)
```

(continues on next page)

(continued from previous page)

```
    Skipping interior check for 1736 sources that fit inside a sphere of radius   53.7 mm
    Skipping solid angle check for 0 points using Qhull
```

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
```

```
220709-17:13:18,262 nipype.workflow INFO:
         [MultiProc] Running 1 tasks, and 0 jobs ready. Free memory (GB): 14.20/14.40,␣
↪Free processors: 0/1.
                     Currently running:
                       * eeg_pipeline.eeg_source_imaging_stage.mne_createfwd
```

```
[Parallel(n_jobs=4)]: Done   2 out of   4 | elapsed:    8.9s remaining:    8.9s
[Parallel(n_jobs=4)]: Done   4 out of   4 | elapsed:    9.0s remaining:    0.0s
[Parallel(n_jobs=4)]: Done   4 out of   4 | elapsed:    9.0s finished
```

```
    Skipping interior check for 1721 sources that fit inside a sphere of radius   53.7 mm
    Skipping solid angle check for 0 points using Qhull
```

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   2 out of   4 | elapsed:    0.3s remaining:    0.3s
[Parallel(n_jobs=4)]: Done   4 out of   4 | elapsed:    0.3s remaining:    0.0s
[Parallel(n_jobs=4)]: Done   4 out of   4 | elapsed:    0.3s finished
```

```

Setting up for EEG...
Computing EEG at 8196 source locations (free orientations)...
```

```
[Parallel(n_jobs=4)]: Using backend LokyBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done   2 out of   4 | elapsed:    1.1s remaining:    1.1s
[Parallel(n_jobs=4)]: Done   4 out of   4 | elapsed:    1.3s remaining:    0.0s
[Parallel(n_jobs=4)]: Done   4 out of   4 | elapsed:    1.3s finished
```

```

Finished.
    Write a source space...
    [done]
    Write a source space...
    [done]
    2 source spaces written
220709-17:13:30,321 nipype.workflow INFO:
         [Node] Finished "mne_createfwd", elapsed time 13.943436s.
220709-17:13:32,276 nipype.workflow INFO:
         [Job 5] Completed (eeg_pipeline.eeg_source_imaging_stage.mne_createfwd).
220709-17:13:32,282 nipype.workflow INFO:
         [MultiProc] Running 0 tasks, and 1 jobs ready. Free memory (GB): 14.40/14.40,␣
↪Free processors: 1/1.
220709-17:13:32,370 nipype.workflow INFO:
         [Node] Setting-up "eeg_pipeline.eeg_source_imaging_stage.mne_invsol" in "/Users/
↪sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/
↪derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_source_imaging_stage/mne_invsol".
220709-17:13:32,379 nipype.workflow INFO:
         [Node] Executing "mne_invsol" <cmtklib.interfaces.mne.MNEInverseSolutionROI>
Reading /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_preprocessing_stage/
↪eeglab2fif/epo.fif ...
    Read a total of 1 projection items:
```

(continues on next page)

(continued from previous page)

```
        Average EEG reference (1 x 128)  idle
    Found the data of interest:
        t =    -200.00 ...     500.00 ms
        0 CTF compensation matrices available
Not setting metadata
Not setting metadata
588 matching events found
No baseline correction applied
Created an SSP operator (subspace dimension = 1)
1 projection items activated
Reading forward solution from /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_
↪pipeline/eeg_source_imaging_stage/mne_createfwd/fwd.fif...
    Reading a source space...
    Computing patch statistics...
    Patch information added...
    Distance information added...
    [done]
    Reading a source space...
    Computing patch statistics...
    Patch information added...
    Distance information added...
    [done]
    2 source spaces read
    Desired named matrix (kind = 3523) not available
    Read EEG forward solution (8196 sources, 128 channels, free orientations)
    Source spaces transformed to the forward solution coordinate frame
    128 x 128 full covariance (kind = 1) found.
    Read a total of 1 projection items:
        Average EEG reference (1 x 128) active
    Reading a source space...
    Computing patch statistics...
    Patch information added...
    Distance information added...
    [done]
    Reading a source space...
    Computing patch statistics...
    Patch information added...
    Distance information added...
    [done]
    2 source spaces read
Computing inverse operator with 128 channels.
    128 out of 128 channels remain after picking
Selected 128 channels
Whitening the forward solution.
    Created an SSP operator (subspace dimension = 1)
Computing rank from covariance with rank=None
    Using tolerance 1.2e-14 (2.2e-16 eps * 128 dim * 0.43  max singular value)
    Estimated rank (eeg): 127
    EEG: rank 127 computed from 128 data channels with 1 projector
    Setting small EEG eigenvalues to zero (without PCA)
Creating the source covariance matrix
```

(continues on next page)

(continued from previous page)

```
Adjusting source covariance matrix.
Computing SVD of whitened and weighted lead field matrix.
220709-17:13:34,277 nipype.workflow INFO:
         [MultiProc] Running 1 tasks, and 0 jobs ready. Free memory (GB): 14.20/14.40,␣
↪Free processors: 0/1.
                     Currently running:
                       * eeg_pipeline.eeg_source_imaging_stage.mne_invsol
    largest singular value = 6.48933
    scaling factor to adjust the trace = 4.66048e+24 (nchan = 128 nzero = 1)
Write inverse operator decomposition in /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_
↪pipeline/eeg_source_imaging_stage/mne_invsol/inv.fif...
    Write a source space...
    [done]
    Write a source space...
    [done]
    2 source spaces written
    Writing inverse operator info...
    Writing noise covariance matrix.
    Writing source covariance matrix.
    Writing orientation priors.
    [done]
Preparing the inverse operator for use...
    Scaled noise and source covariance from nave = 1 to nave = 588
    Created the regularized inverter
    Created an SSP operator (subspace dimension = 1)
    Created the whitener using a noise covariance matrix with rank 127 (1 small␣
↪eigenvalues omitted)
    Computing noise-normalization factors (sLORETA)...
[done]
Picked 128 channels from the data
Computing inverse...
    Eigenleads need to be weighted ...
Processing epoch : 1 / 588
combining the current components...
Processing epoch : 2 / 588
combining the current components...
Processing epoch : 3 / 588
combining the current components...
Processing epoch : 4 / 588
combining the current components...
Processing epoch : 5 / 588
combining the current components...
Processing epoch : 6 / 588
combining the current components...
Processing epoch : 7 / 588
combining the current components...
Processing epoch : 8 / 588
combining the current components...
Processing epoch : 9 / 588
combining the current components...
Processing epoch : 10 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 11 / 588
combining the current components...
Processing epoch : 12 / 588
combining the current components...
Processing epoch : 13 / 588
combining the current components...
Processing epoch : 14 / 588
combining the current components...
Processing epoch : 15 / 588
combining the current components...
Processing epoch : 16 / 588
combining the current components...
Processing epoch : 17 / 588
combining the current components...
Processing epoch : 18 / 588
combining the current components...
Processing epoch : 19 / 588
combining the current components...
Processing epoch : 20 / 588
combining the current components...
Processing epoch : 21 / 588
combining the current components...
Processing epoch : 22 / 588
combining the current components...
Processing epoch : 23 / 588
combining the current components...
Processing epoch : 24 / 588
combining the current components...
Processing epoch : 25 / 588
combining the current components...
Processing epoch : 26 / 588
combining the current components...
Processing epoch : 27 / 588
combining the current components...
Processing epoch : 28 / 588
combining the current components...
Processing epoch : 29 / 588
combining the current components...
Processing epoch : 30 / 588
combining the current components...
Processing epoch : 31 / 588
combining the current components...
Processing epoch : 32 / 588
combining the current components...
Processing epoch : 33 / 588
combining the current components...
Processing epoch : 34 / 588
combining the current components...
Processing epoch : 35 / 588
combining the current components...
Processing epoch : 36 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 37 / 588
combining the current components...
Processing epoch : 38 / 588
combining the current components...
Processing epoch : 39 / 588
combining the current components...
Processing epoch : 40 / 588
combining the current components...
Processing epoch : 41 / 588
combining the current components...
Processing epoch : 42 / 588
combining the current components...
Processing epoch : 43 / 588
combining the current components...
Processing epoch : 44 / 588
combining the current components...
Processing epoch : 45 / 588
combining the current components...
Processing epoch : 46 / 588
combining the current components...
Processing epoch : 47 / 588
combining the current components...
Processing epoch : 48 / 588
combining the current components...
Processing epoch : 49 / 588
combining the current components...
Processing epoch : 50 / 588
combining the current components...
Processing epoch : 51 / 588
combining the current components...
Processing epoch : 52 / 588
combining the current components...
Processing epoch : 53 / 588
combining the current components...
Processing epoch : 54 / 588
combining the current components...
Processing epoch : 55 / 588
combining the current components...
Processing epoch : 56 / 588
combining the current components...
Processing epoch : 57 / 588
combining the current components...
Processing epoch : 58 / 588
combining the current components...
Processing epoch : 59 / 588
combining the current components...
Processing epoch : 60 / 588
combining the current components...
Processing epoch : 61 / 588
combining the current components...
Processing epoch : 62 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 63 / 588
combining the current components...
Processing epoch : 64 / 588
combining the current components...
Processing epoch : 65 / 588
combining the current components...
Processing epoch : 66 / 588
combining the current components...
Processing epoch : 67 / 588
combining the current components...
Processing epoch : 68 / 588
combining the current components...
Processing epoch : 69 / 588
combining the current components...
Processing epoch : 70 / 588
combining the current components...
Processing epoch : 71 / 588
combining the current components...
Processing epoch : 72 / 588
combining the current components...
Processing epoch : 73 / 588
combining the current components...
Processing epoch : 74 / 588
combining the current components...
Processing epoch : 75 / 588
combining the current components...
Processing epoch : 76 / 588
combining the current components...
Processing epoch : 77 / 588
combining the current components...
Processing epoch : 78 / 588
combining the current components...
Processing epoch : 79 / 588
combining the current components...
Processing epoch : 80 / 588
combining the current components...
Processing epoch : 81 / 588
combining the current components...
Processing epoch : 82 / 588
combining the current components...
Processing epoch : 83 / 588
combining the current components...
Processing epoch : 84 / 588
combining the current components...
Processing epoch : 85 / 588
combining the current components...
Processing epoch : 86 / 588
combining the current components...
Processing epoch : 87 / 588
combining the current components...
Processing epoch : 88 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 89 / 588
combining the current components...
Processing epoch : 90 / 588
combining the current components...
Processing epoch : 91 / 588
combining the current components...
Processing epoch : 92 / 588
combining the current components...
Processing epoch : 93 / 588
combining the current components...
Processing epoch : 94 / 588
combining the current components...
Processing epoch : 95 / 588
combining the current components...
Processing epoch : 96 / 588
combining the current components...
Processing epoch : 97 / 588
combining the current components...
Processing epoch : 98 / 588
combining the current components...
Processing epoch : 99 / 588
combining the current components...
Processing epoch : 100 / 588
combining the current components...
Processing epoch : 101 / 588
combining the current components...
Processing epoch : 102 / 588
combining the current components...
Processing epoch : 103 / 588
combining the current components...
Processing epoch : 104 / 588
combining the current components...
Processing epoch : 105 / 588
combining the current components...
Processing epoch : 106 / 588
combining the current components...
Processing epoch : 107 / 588
combining the current components...
Processing epoch : 108 / 588
combining the current components...
Processing epoch : 109 / 588
combining the current components...
Processing epoch : 110 / 588
combining the current components...
Processing epoch : 111 / 588
combining the current components...
Processing epoch : 112 / 588
combining the current components...
Processing epoch : 113 / 588
combining the current components...
Processing epoch : 114 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 115 / 588
combining the current components...
Processing epoch : 116 / 588
combining the current components...
Processing epoch : 117 / 588
combining the current components...
Processing epoch : 118 / 588
combining the current components...
Processing epoch : 119 / 588
combining the current components...
Processing epoch : 120 / 588
combining the current components...
Processing epoch : 121 / 588
combining the current components...
Processing epoch : 122 / 588
combining the current components...
Processing epoch : 123 / 588
combining the current components...
Processing epoch : 124 / 588
combining the current components...
Processing epoch : 125 / 588
combining the current components...
Processing epoch : 126 / 588
combining the current components...
Processing epoch : 127 / 588
combining the current components...
Processing epoch : 128 / 588
combining the current components...
Processing epoch : 129 / 588
combining the current components...
Processing epoch : 130 / 588
combining the current components...
Processing epoch : 131 / 588
combining the current components...
Processing epoch : 132 / 588
combining the current components...
Processing epoch : 133 / 588
combining the current components...
Processing epoch : 134 / 588
combining the current components...
Processing epoch : 135 / 588
combining the current components...
Processing epoch : 136 / 588
combining the current components...
Processing epoch : 137 / 588
combining the current components...
Processing epoch : 138 / 588
combining the current components...
Processing epoch : 139 / 588
combining the current components...
Processing epoch : 140 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 141 / 588
combining the current components...
Processing epoch : 142 / 588
combining the current components...
Processing epoch : 143 / 588
combining the current components...
Processing epoch : 144 / 588
combining the current components...
Processing epoch : 145 / 588
combining the current components...
Processing epoch : 146 / 588
combining the current components...
Processing epoch : 147 / 588
combining the current components...
Processing epoch : 148 / 588
combining the current components...
Processing epoch : 149 / 588
combining the current components...
Processing epoch : 150 / 588
combining the current components...
Processing epoch : 151 / 588
combining the current components...
Processing epoch : 152 / 588
combining the current components...
Processing epoch : 153 / 588
combining the current components...
Processing epoch : 154 / 588
combining the current components...
Processing epoch : 155 / 588
combining the current components...
Processing epoch : 156 / 588
combining the current components...
Processing epoch : 157 / 588
combining the current components...
Processing epoch : 158 / 588
combining the current components...
Processing epoch : 159 / 588
combining the current components...
Processing epoch : 160 / 588
combining the current components...
Processing epoch : 161 / 588
combining the current components...
Processing epoch : 162 / 588
combining the current components...
Processing epoch : 163 / 588
combining the current components...
Processing epoch : 164 / 588
combining the current components...
Processing epoch : 165 / 588
combining the current components...
Processing epoch : 166 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 167 / 588
combining the current components...
Processing epoch : 168 / 588
combining the current components...
Processing epoch : 169 / 588
combining the current components...
Processing epoch : 170 / 588
combining the current components...
Processing epoch : 171 / 588
combining the current components...
Processing epoch : 172 / 588
combining the current components...
Processing epoch : 173 / 588
combining the current components...
Processing epoch : 174 / 588
combining the current components...
Processing epoch : 175 / 588
combining the current components...
Processing epoch : 176 / 588
combining the current components...
Processing epoch : 177 / 588
combining the current components...
Processing epoch : 178 / 588
combining the current components...
Processing epoch : 179 / 588
combining the current components...
Processing epoch : 180 / 588
combining the current components...
Processing epoch : 181 / 588
combining the current components...
Processing epoch : 182 / 588
combining the current components...
Processing epoch : 183 / 588
combining the current components...
Processing epoch : 184 / 588
combining the current components...
Processing epoch : 185 / 588
combining the current components...
Processing epoch : 186 / 588
combining the current components...
Processing epoch : 187 / 588
combining the current components...
Processing epoch : 188 / 588
combining the current components...
Processing epoch : 189 / 588
combining the current components...
Processing epoch : 190 / 588
combining the current components...
Processing epoch : 191 / 588
combining the current components...
Processing epoch : 192 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 193 / 588
combining the current components...
Processing epoch : 194 / 588
combining the current components...
Processing epoch : 195 / 588
combining the current components...
Processing epoch : 196 / 588
combining the current components...
Processing epoch : 197 / 588
combining the current components...
Processing epoch : 198 / 588
combining the current components...
Processing epoch : 199 / 588
combining the current components...
Processing epoch : 200 / 588
combining the current components...
Processing epoch : 201 / 588
combining the current components...
Processing epoch : 202 / 588
combining the current components...
Processing epoch : 203 / 588
combining the current components...
Processing epoch : 204 / 588
combining the current components...
Processing epoch : 205 / 588
combining the current components...
Processing epoch : 206 / 588
combining the current components...
Processing epoch : 207 / 588
combining the current components...
Processing epoch : 208 / 588
combining the current components...
Processing epoch : 209 / 588
combining the current components...
Processing epoch : 210 / 588
combining the current components...
Processing epoch : 211 / 588
combining the current components...
Processing epoch : 212 / 588
combining the current components...
Processing epoch : 213 / 588
combining the current components...
Processing epoch : 214 / 588
combining the current components...
Processing epoch : 215 / 588
combining the current components...
Processing epoch : 216 / 588
combining the current components...
Processing epoch : 217 / 588
combining the current components...
Processing epoch : 218 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 219 / 588
combining the current components...
Processing epoch : 220 / 588
combining the current components...
Processing epoch : 221 / 588
combining the current components...
Processing epoch : 222 / 588
combining the current components...
Processing epoch : 223 / 588
combining the current components...
Processing epoch : 224 / 588
combining the current components...
Processing epoch : 225 / 588
combining the current components...
Processing epoch : 226 / 588
combining the current components...
Processing epoch : 227 / 588
combining the current components...
Processing epoch : 228 / 588
combining the current components...
Processing epoch : 229 / 588
combining the current components...
Processing epoch : 230 / 588
combining the current components...
Processing epoch : 231 / 588
combining the current components...
Processing epoch : 232 / 588
combining the current components...
Processing epoch : 233 / 588
combining the current components...
Processing epoch : 234 / 588
combining the current components...
Processing epoch : 235 / 588
combining the current components...
Processing epoch : 236 / 588
combining the current components...
Processing epoch : 237 / 588
combining the current components...
Processing epoch : 238 / 588
combining the current components...
Processing epoch : 239 / 588
combining the current components...
Processing epoch : 240 / 588
combining the current components...
Processing epoch : 241 / 588
combining the current components...
Processing epoch : 242 / 588
combining the current components...
Processing epoch : 243 / 588
combining the current components...
Processing epoch : 244 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 245 / 588
combining the current components...
Processing epoch : 246 / 588
combining the current components...
Processing epoch : 247 / 588
combining the current components...
Processing epoch : 248 / 588
combining the current components...
Processing epoch : 249 / 588
combining the current components...
Processing epoch : 250 / 588
combining the current components...
Processing epoch : 251 / 588
combining the current components...
Processing epoch : 252 / 588
combining the current components...
Processing epoch : 253 / 588
combining the current components...
Processing epoch : 254 / 588
combining the current components...
Processing epoch : 255 / 588
combining the current components...
Processing epoch : 256 / 588
combining the current components...
Processing epoch : 257 / 588
combining the current components...
Processing epoch : 258 / 588
combining the current components...
Processing epoch : 259 / 588
combining the current components...
Processing epoch : 260 / 588
combining the current components...
Processing epoch : 261 / 588
combining the current components...
Processing epoch : 262 / 588
combining the current components...
Processing epoch : 263 / 588
combining the current components...
Processing epoch : 264 / 588
combining the current components...
Processing epoch : 265 / 588
combining the current components...
Processing epoch : 266 / 588
combining the current components...
Processing epoch : 267 / 588
combining the current components...
Processing epoch : 268 / 588
combining the current components...
Processing epoch : 269 / 588
combining the current components...
Processing epoch : 270 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 271 / 588
combining the current components...
Processing epoch : 272 / 588
combining the current components...
Processing epoch : 273 / 588
combining the current components...
Processing epoch : 274 / 588
combining the current components...
Processing epoch : 275 / 588
combining the current components...
Processing epoch : 276 / 588
combining the current components...
Processing epoch : 277 / 588
combining the current components...
Processing epoch : 278 / 588
combining the current components...
Processing epoch : 279 / 588
combining the current components...
Processing epoch : 280 / 588
combining the current components...
Processing epoch : 281 / 588
combining the current components...
Processing epoch : 282 / 588
combining the current components...
Processing epoch : 283 / 588
combining the current components...
Processing epoch : 284 / 588
combining the current components...
Processing epoch : 285 / 588
combining the current components...
Processing epoch : 286 / 588
combining the current components...
Processing epoch : 287 / 588
combining the current components...
Processing epoch : 288 / 588
combining the current components...
Processing epoch : 289 / 588
combining the current components...
Processing epoch : 290 / 588
combining the current components...
Processing epoch : 291 / 588
combining the current components...
Processing epoch : 292 / 588
combining the current components...
Processing epoch : 293 / 588
combining the current components...
Processing epoch : 294 / 588
combining the current components...
Processing epoch : 295 / 588
combining the current components...
Processing epoch : 296 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 297 / 588
combining the current components...
Processing epoch : 298 / 588
combining the current components...
Processing epoch : 299 / 588
combining the current components...
Processing epoch : 300 / 588
combining the current components...
Processing epoch : 301 / 588
combining the current components...
Processing epoch : 302 / 588
combining the current components...
Processing epoch : 303 / 588
combining the current components...
Processing epoch : 304 / 588
combining the current components...
Processing epoch : 305 / 588
combining the current components...
Processing epoch : 306 / 588
combining the current components...
Processing epoch : 307 / 588
combining the current components...
Processing epoch : 308 / 588
combining the current components...
Processing epoch : 309 / 588
combining the current components...
Processing epoch : 310 / 588
combining the current components...
Processing epoch : 311 / 588
combining the current components...
Processing epoch : 312 / 588
combining the current components...
Processing epoch : 313 / 588
combining the current components...
Processing epoch : 314 / 588
combining the current components...
Processing epoch : 315 / 588
combining the current components...
Processing epoch : 316 / 588
combining the current components...
Processing epoch : 317 / 588
combining the current components...
Processing epoch : 318 / 588
combining the current components...
Processing epoch : 319 / 588
combining the current components...
Processing epoch : 320 / 588
combining the current components...
Processing epoch : 321 / 588
combining the current components...
Processing epoch : 322 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 323 / 588
combining the current components...
Processing epoch : 324 / 588
combining the current components...
Processing epoch : 325 / 588
combining the current components...
Processing epoch : 326 / 588
combining the current components...
Processing epoch : 327 / 588
combining the current components...
Processing epoch : 328 / 588
combining the current components...
Processing epoch : 329 / 588
combining the current components...
Processing epoch : 330 / 588
combining the current components...
Processing epoch : 331 / 588
combining the current components...
Processing epoch : 332 / 588
combining the current components...
Processing epoch : 333 / 588
combining the current components...
Processing epoch : 334 / 588
combining the current components...
Processing epoch : 335 / 588
combining the current components...
Processing epoch : 336 / 588
combining the current components...
Processing epoch : 337 / 588
combining the current components...
Processing epoch : 338 / 588
combining the current components...
Processing epoch : 339 / 588
combining the current components...
Processing epoch : 340 / 588
combining the current components...
Processing epoch : 341 / 588
combining the current components...
Processing epoch : 342 / 588
combining the current components...
Processing epoch : 343 / 588
combining the current components...
Processing epoch : 344 / 588
combining the current components...
Processing epoch : 345 / 588
combining the current components...
Processing epoch : 346 / 588
combining the current components...
Processing epoch : 347 / 588
combining the current components...
Processing epoch : 348 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 349 / 588
combining the current components...
Processing epoch : 350 / 588
combining the current components...
Processing epoch : 351 / 588
combining the current components...
Processing epoch : 352 / 588
combining the current components...
Processing epoch : 353 / 588
combining the current components...
Processing epoch : 354 / 588
combining the current components...
Processing epoch : 355 / 588
combining the current components...
Processing epoch : 356 / 588
combining the current components...
Processing epoch : 357 / 588
combining the current components...
Processing epoch : 358 / 588
combining the current components...
Processing epoch : 359 / 588
combining the current components...
Processing epoch : 360 / 588
combining the current components...
Processing epoch : 361 / 588
combining the current components...
Processing epoch : 362 / 588
combining the current components...
Processing epoch : 363 / 588
combining the current components...
Processing epoch : 364 / 588
combining the current components...
Processing epoch : 365 / 588
combining the current components...
Processing epoch : 366 / 588
combining the current components...
Processing epoch : 367 / 588
combining the current components...
Processing epoch : 368 / 588
combining the current components...
Processing epoch : 369 / 588
combining the current components...
Processing epoch : 370 / 588
combining the current components...
Processing epoch : 371 / 588
combining the current components...
Processing epoch : 372 / 588
combining the current components...
Processing epoch : 373 / 588
combining the current components...
Processing epoch : 374 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 375 / 588
combining the current components...
Processing epoch : 376 / 588
combining the current components...
Processing epoch : 377 / 588
combining the current components...
Processing epoch : 378 / 588
combining the current components...
Processing epoch : 379 / 588
combining the current components...
Processing epoch : 380 / 588
combining the current components...
Processing epoch : 381 / 588
combining the current components...
Processing epoch : 382 / 588
combining the current components...
Processing epoch : 383 / 588
combining the current components...
Processing epoch : 384 / 588
combining the current components...
Processing epoch : 385 / 588
combining the current components...
Processing epoch : 386 / 588
combining the current components...
Processing epoch : 387 / 588
combining the current components...
Processing epoch : 388 / 588
combining the current components...
Processing epoch : 389 / 588
combining the current components...
Processing epoch : 390 / 588
combining the current components...
Processing epoch : 391 / 588
combining the current components...
Processing epoch : 392 / 588
combining the current components...
Processing epoch : 393 / 588
combining the current components...
Processing epoch : 394 / 588
combining the current components...
Processing epoch : 395 / 588
combining the current components...
Processing epoch : 396 / 588
combining the current components...
Processing epoch : 397 / 588
combining the current components...
Processing epoch : 398 / 588
combining the current components...
Processing epoch : 399 / 588
combining the current components...
Processing epoch : 400 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 401 / 588
combining the current components...
Processing epoch : 402 / 588
combining the current components...
Processing epoch : 403 / 588
combining the current components...
Processing epoch : 404 / 588
combining the current components...
Processing epoch : 405 / 588
combining the current components...
Processing epoch : 406 / 588
combining the current components...
Processing epoch : 407 / 588
combining the current components...
Processing epoch : 408 / 588
combining the current components...
Processing epoch : 409 / 588
combining the current components...
Processing epoch : 410 / 588
combining the current components...
Processing epoch : 411 / 588
combining the current components...
Processing epoch : 412 / 588
combining the current components...
Processing epoch : 413 / 588
combining the current components...
Processing epoch : 414 / 588
combining the current components...
Processing epoch : 415 / 588
combining the current components...
Processing epoch : 416 / 588
combining the current components...
Processing epoch : 417 / 588
combining the current components...
Processing epoch : 418 / 588
combining the current components...
Processing epoch : 419 / 588
combining the current components...
Processing epoch : 420 / 588
combining the current components...
Processing epoch : 421 / 588
combining the current components...
Processing epoch : 422 / 588
combining the current components...
Processing epoch : 423 / 588
combining the current components...
Processing epoch : 424 / 588
combining the current components...
Processing epoch : 425 / 588
combining the current components...
Processing epoch : 426 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 427 / 588
combining the current components...
Processing epoch : 428 / 588
combining the current components...
Processing epoch : 429 / 588
combining the current components...
Processing epoch : 430 / 588
combining the current components...
Processing epoch : 431 / 588
combining the current components...
Processing epoch : 432 / 588
combining the current components...
Processing epoch : 433 / 588
combining the current components...
Processing epoch : 434 / 588
combining the current components...
Processing epoch : 435 / 588
combining the current components...
Processing epoch : 436 / 588
combining the current components...
Processing epoch : 437 / 588
combining the current components...
Processing epoch : 438 / 588
combining the current components...
Processing epoch : 439 / 588
combining the current components...
Processing epoch : 440 / 588
combining the current components...
Processing epoch : 441 / 588
combining the current components...
Processing epoch : 442 / 588
combining the current components...
Processing epoch : 443 / 588
combining the current components...
Processing epoch : 444 / 588
combining the current components...
Processing epoch : 445 / 588
combining the current components...
Processing epoch : 446 / 588
combining the current components...
Processing epoch : 447 / 588
combining the current components...
Processing epoch : 448 / 588
combining the current components...
Processing epoch : 449 / 588
combining the current components...
Processing epoch : 450 / 588
combining the current components...
Processing epoch : 451 / 588
combining the current components...
Processing epoch : 452 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 453 / 588
combining the current components...
Processing epoch : 454 / 588
combining the current components...
Processing epoch : 455 / 588
combining the current components...
Processing epoch : 456 / 588
combining the current components...
Processing epoch : 457 / 588
combining the current components...
Processing epoch : 458 / 588
combining the current components...
Processing epoch : 459 / 588
combining the current components...
Processing epoch : 460 / 588
combining the current components...
Processing epoch : 461 / 588
combining the current components...
Processing epoch : 462 / 588
combining the current components...
Processing epoch : 463 / 588
combining the current components...
Processing epoch : 464 / 588
combining the current components...
Processing epoch : 465 / 588
combining the current components...
Processing epoch : 466 / 588
combining the current components...
Processing epoch : 467 / 588
combining the current components...
Processing epoch : 468 / 588
combining the current components...
Processing epoch : 469 / 588
combining the current components...
Processing epoch : 470 / 588
combining the current components...
Processing epoch : 471 / 588
combining the current components...
Processing epoch : 472 / 588
combining the current components...
Processing epoch : 473 / 588
combining the current components...
Processing epoch : 474 / 588
combining the current components...
Processing epoch : 475 / 588
combining the current components...
Processing epoch : 476 / 588
combining the current components...
Processing epoch : 477 / 588
combining the current components...
Processing epoch : 478 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 479 / 588
combining the current components...
Processing epoch : 480 / 588
combining the current components...
Processing epoch : 481 / 588
combining the current components...
Processing epoch : 482 / 588
combining the current components...
Processing epoch : 483 / 588
combining the current components...
Processing epoch : 484 / 588
combining the current components...
Processing epoch : 485 / 588
combining the current components...
Processing epoch : 486 / 588
combining the current components...
Processing epoch : 487 / 588
combining the current components...
Processing epoch : 488 / 588
combining the current components...
Processing epoch : 489 / 588
combining the current components...
Processing epoch : 490 / 588
combining the current components...
Processing epoch : 491 / 588
combining the current components...
Processing epoch : 492 / 588
combining the current components...
Processing epoch : 493 / 588
combining the current components...
Processing epoch : 494 / 588
combining the current components...
Processing epoch : 495 / 588
combining the current components...
Processing epoch : 496 / 588
combining the current components...
Processing epoch : 497 / 588
combining the current components...
Processing epoch : 498 / 588
combining the current components...
Processing epoch : 499 / 588
combining the current components...
Processing epoch : 500 / 588
combining the current components...
Processing epoch : 501 / 588
combining the current components...
Processing epoch : 502 / 588
combining the current components...
Processing epoch : 503 / 588
combining the current components...
Processing epoch : 504 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 505 / 588
combining the current components...
Processing epoch : 506 / 588
combining the current components...
Processing epoch : 507 / 588
combining the current components...
Processing epoch : 508 / 588
combining the current components...
Processing epoch : 509 / 588
combining the current components...
Processing epoch : 510 / 588
combining the current components...
Processing epoch : 511 / 588
combining the current components...
Processing epoch : 512 / 588
combining the current components...
Processing epoch : 513 / 588
combining the current components...
Processing epoch : 514 / 588
combining the current components...
Processing epoch : 515 / 588
combining the current components...
Processing epoch : 516 / 588
combining the current components...
Processing epoch : 517 / 588
combining the current components...
Processing epoch : 518 / 588
combining the current components...
Processing epoch : 519 / 588
combining the current components...
Processing epoch : 520 / 588
combining the current components...
Processing epoch : 521 / 588
combining the current components...
Processing epoch : 522 / 588
combining the current components...
Processing epoch : 523 / 588
combining the current components...
Processing epoch : 524 / 588
combining the current components...
Processing epoch : 525 / 588
combining the current components...
Processing epoch : 526 / 588
combining the current components...
Processing epoch : 527 / 588
combining the current components...
Processing epoch : 528 / 588
combining the current components...
Processing epoch : 529 / 588
combining the current components...
Processing epoch : 530 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 531 / 588
combining the current components...
Processing epoch : 532 / 588
combining the current components...
Processing epoch : 533 / 588
combining the current components...
Processing epoch : 534 / 588
combining the current components...
Processing epoch : 535 / 588
combining the current components...
Processing epoch : 536 / 588
combining the current components...
Processing epoch : 537 / 588
combining the current components...
Processing epoch : 538 / 588
combining the current components...
Processing epoch : 539 / 588
combining the current components...
Processing epoch : 540 / 588
combining the current components...
Processing epoch : 541 / 588
combining the current components...
Processing epoch : 542 / 588
combining the current components...
Processing epoch : 543 / 588
combining the current components...
Processing epoch : 544 / 588
combining the current components...
Processing epoch : 545 / 588
combining the current components...
Processing epoch : 546 / 588
combining the current components...
Processing epoch : 547 / 588
combining the current components...
Processing epoch : 548 / 588
combining the current components...
Processing epoch : 549 / 588
combining the current components...
Processing epoch : 550 / 588
combining the current components...
Processing epoch : 551 / 588
combining the current components...
Processing epoch : 552 / 588
combining the current components...
Processing epoch : 553 / 588
combining the current components...
Processing epoch : 554 / 588
combining the current components...
Processing epoch : 555 / 588
combining the current components...
Processing epoch : 556 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 557 / 588
combining the current components...
Processing epoch : 558 / 588
combining the current components...
Processing epoch : 559 / 588
combining the current components...
Processing epoch : 560 / 588
combining the current components...
Processing epoch : 561 / 588
combining the current components...
Processing epoch : 562 / 588
combining the current components...
Processing epoch : 563 / 588
combining the current components...
Processing epoch : 564 / 588
combining the current components...
Processing epoch : 565 / 588
combining the current components...
Processing epoch : 566 / 588
combining the current components...
Processing epoch : 567 / 588
combining the current components...
Processing epoch : 568 / 588
combining the current components...
Processing epoch : 569 / 588
combining the current components...
Processing epoch : 570 / 588
combining the current components...
Processing epoch : 571 / 588
combining the current components...
Processing epoch : 572 / 588
combining the current components...
Processing epoch : 573 / 588
combining the current components...
Processing epoch : 574 / 588
combining the current components...
Processing epoch : 575 / 588
combining the current components...
Processing epoch : 576 / 588
combining the current components...
Processing epoch : 577 / 588
combining the current components...
Processing epoch : 578 / 588
combining the current components...
Processing epoch : 579 / 588
combining the current components...
Processing epoch : 580 / 588
combining the current components...
Processing epoch : 581 / 588
combining the current components...
Processing epoch : 582 / 588
```

(continues on next page)

(continued from previous page)

```
combining the current components...
Processing epoch : 583 / 588
combining the current components...
Processing epoch : 584 / 588
combining the current components...
Processing epoch : 585 / 588
combining the current components...
Processing epoch : 586 / 588
combining the current components...
Processing epoch : 587 / 588
combining the current components...
Processing epoch : 588 / 588
combining the current components...
[done]
Reading labels from parcellation...
   read 35 labels from /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
→notebooks/ds003505_demo/derivatives/freesurfer-7.1.1/sub-01/label/lh.lausanne2018.
→scale1.annot
   read 35 labels from /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
→notebooks/ds003505_demo/derivatives/freesurfer-7.1.1/sub-01/label/rh.lausanne2018.
→scale1.annot
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
```

(continues on next page)

(continued from previous page)

```
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
```

(continues on next page)

(continued from previous page)

```
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
```

(continues on next page)

(continued from previous page)

```
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
```

(continues on next page)

(continued from previous page)

```
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
```

(continues on next page)

(continued from previous page)

```
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
```

(continues on next page)

(continued from previous page)

```
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
```

(continues on next page)

(continued from previous page)

```
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
```

(continues on next page)

(continued from previous page)

```
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
```

(continues on next page)

(continued from previous page)

```
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
```

(continues on next page)

(continued from previous page)

```
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
```

(continues on next page)

(continued from previous page)

```
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
Extracting time courses for 70 labels (mode: pca_flip)
220709-17:16:19,251 nipype.workflow INFO:
         [Node] Finished "mne_invsol", elapsed time 166.864983s.
220709-17:16:20,461 nipype.workflow INFO:
         [Job 6] Completed (eeg_pipeline.eeg_source_imaging_stage.mne_invsol).
220709-17:16:20,473 nipype.workflow INFO:
         [MultiProc] Running 0 tasks, and 1 jobs ready. Free memory (GB): 14.40/14.40,␣
↪Free processors: 1/1.
220709-17:16:20,889 nipype.workflow INFO:
         [Node] Setting-up "eeg_pipeline.eeg_connectome_stage.eeg_compute_matrice" in "/
↪Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_connectome_stage/eeg_compute_
↪matrice".
220709-17:16:20,901 nipype.workflow INFO:
         [Node] Executing "eeg_compute_matrice" <cmtklib.interfaces.mne.
↪MNESpectralConnectivity>
```

(continues on next page)

(continued from previous page)

```
Reading /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_preprocessing_stage/
↪eeglab2fif/epo.fif ...
    Read a total of 1 projection items:
        Average EEG reference (1 x 128)  idle
    Found the data of interest:
        t =    -200.00 ...     500.00 ms
        0 CTF compensation matrices available
Not setting metadata
Not setting metadata
588 matching events found
No baseline correction applied
Created an SSP operator (subspace dimension = 1)
1 projection items activated
220709-17:16:22,464 nipype.workflow INFO:
         [MultiProc] Running 1 tasks, and 0 jobs ready. Free memory (GB): 14.20/14.40,␣
↪Free processors: 0/1.
                     Currently running:
                       * eeg_pipeline.eeg_connectome_stage.eeg_compute_matrice
Save /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_connectome_stage/eeg_compute_
↪matrice/conndata-network_connectivity.tsv...
Save /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_connectome_stage/eeg_compute_
↪matrice/conndata-network_connectivity.gpickle...
Save /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_connectome_stage/eeg_compute_
↪matrice/conndata-network_connectivity.mat...
Save /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/nipype-1.7.0/sub-01/eeg_pipeline/eeg_connectome_stage/eeg_compute_
↪matrice/conndata-network_connectivity.graphml...
220709-17:17:14,972 nipype.workflow INFO:
         [Node] Finished "eeg_compute_matrice", elapsed time 54.067281s.
220709-17:17:16,563 nipype.workflow INFO:
         [Job 7] Completed (eeg_pipeline.eeg_connectome_stage.eeg_compute_matrice).
220709-17:17:16,570 nipype.workflow INFO:
         [MultiProc] Running 0 tasks, and 1 jobs ready. Free memory (GB): 14.40/14.40,␣
↪Free processors: 1/1.
220709-17:17:16,685 nipype.workflow INFO:
         [Node] Setting-up "eeg_pipeline.eeg_datasinker" in "/Users/sebastientourbier/
↪Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/derivatives/nipype-1.7.
↪0/sub-01/eeg_pipeline/eeg_datasinker".
220709-17:17:16,707 nipype.workflow INFO:
         [Node] Executing "eeg_datasinker" <nipype.interfaces.io.DataSink>
220709-17:17:16,711 nipype.interface INFO:
         sub: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/epo.fif -> /Users/sebastientourbier/
↪Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.1.0/
↪sub-01/eeg/sub-01_task-faces_epo.fif
220709-17:17:16,716 nipype.interface INFO:
         sub: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/bem.fif -> /Users/sebastientourbier/
↪Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.1.0/
↪sub-01/eeg/sub-01_task-faces_bem.fif
```

(continues on next page)

(continued from previous page)

```
220709-17:17:16,720 nipype.interface INFO:
         sub: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/noisecov.fif -> /Users/
↪sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/
↪derivatives/cmp-v3.1.0/sub-01/eeg/sub-01_task-faces_noisecov.fif
220709-17:17:16,724 nipype.interface INFO:
         sub: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/src.fif -> /Users/sebastientourbier/
↪Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.1.0/
↪sub-01/eeg/sub-01_task-faces_src.fif
220709-17:17:16,727 nipype.interface INFO:
         sub: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/fwd.fif -> /Users/sebastientourbier/
↪Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.1.0/
↪sub-01/eeg/sub-01_task-faces_fwd.fif
220709-17:17:16,731 nipype.interface INFO:
         sub: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/inv.fif -> /Users/sebastientourbier/
↪Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.1.0/
↪sub-01/eeg/sub-01_task-faces_inv.fif
220709-17:17:16,734 nipype.interface INFO:
         sub: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/timeseries.pickle -> /Users/
↪sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_demo/
↪derivatives/cmp-v3.1.0/sub-01/eeg/sub-01_task-faces_atlas-L2018_res-scale1_timeseries.
↪pickle
220709-17:17:16,736 nipype.interface INFO:
         sub: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/conndata-network_connectivity.tsv -> /
↪Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/cmp-v3.1.0/sub-01/eeg/sub-01_task-faces_atlas-L2018_res-scale1_
↪conndata-network_connectivity.tsv
220709-17:17:16,739 nipype.interface INFO:
         sub: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/conndata-network_connectivity.gpickle -
↪> /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/cmp-v3.1.0/sub-01/eeg/sub-01_task-faces_atlas-L2018_res-scale1_
↪conndata-network_connectivity.gpickle
220709-17:17:16,742 nipype.interface INFO:
         sub: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/conndata-network_connectivity.mat -> /
↪Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/cmp-v3.1.0/sub-01/eeg/sub-01_task-faces_atlas-L2018_res-scale1_
↪conndata-network_connectivity.mat
220709-17:17:16,745 nipype.interface INFO:
         sub: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/conndata-network_connectivity.graphml -
↪> /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/ds003505_
↪demo/derivatives/cmp-v3.1.0/sub-01/eeg/sub-01_task-faces_atlas-L2018_res-scale1_
↪conndata-network_connectivity.graphml
220709-17:17:16,749 nipype.workflow INFO:
         [Node] Finished "eeg_datasinker", elapsed time 0.037833s.
```

(continues on next page)

(continued from previous page)

```
220709-17:17:18,568 nipype.workflow INFO:
         [Job 8] Completed (eeg_pipeline.eeg_datasinker).
220709-17:17:18,576 nipype.workflow INFO:
         [MultiProc] Running 0 tasks, and 0 jobs ready. Free memory (GB): 14.40/14.40,␣
↪Free processors: 1/1.
220709-17:17:21,115 nipype.interface INFO:
         **** Processing finished ****
CPU times: user 16.5 s, sys: 1.98 s, total: 18.4 s
Wall time: 15min 58s
```

```
/Applications/miniconda3/envs/py37cmp-eeg/lib/python3.7/site-packages/joblib/externals/
↪loky/backend/resource_tracker.py:320: UserWarning: resource_tracker: There appear to␣
↪be 6 leaked folder objects to clean up at shutdown
  (len(rtype_registry), rtype))
/Applications/miniconda3/envs/py37cmp-eeg/lib/python3.7/site-packages/joblib/externals/
↪loky/backend/resource_tracker.py:333: UserWarning: resource_tracker: /var/folders/vy/
↪0bw_1jvj54n8lvcgvdrtqb0c0000gn/T/joblib_memmapping_folder_47551_
↪379a43977f814190bfe6f03e92cfdae6_3fedd64ce92b49b9a6481229dccca45d: FileNotFoundError(2,
↪ 'No such file or directory')
  warnings.warn('resource_tracker: %s: %r' % (name, e))
/Applications/miniconda3/envs/py37cmp-eeg/lib/python3.7/site-packages/joblib/externals/
↪loky/backend/resource_tracker.py:333: UserWarning: resource_tracker: /var/folders/vy/
↪0bw_1jvj54n8lvcgvdrtqb0c0000gn/T/joblib_memmapping_folder_47551_
↪379a43977f814190bfe6f03e92cfdae6_ae6b9e5318df4c718513b5ce780ef199: FileNotFoundError(2,
↪ 'No such file or directory')
  warnings.warn('resource_tracker: %s: %r' % (name, e))
/Applications/miniconda3/envs/py37cmp-eeg/lib/python3.7/site-packages/joblib/externals/
↪loky/backend/resource_tracker.py:333: UserWarning: resource_tracker: /var/folders/vy/
↪0bw_1jvj54n8lvcgvdrtqb0c0000gn/T/joblib_memmapping_folder_47551_
↪3d28f25931e34ead98c3102ee1bbe4be_e207a49f2faf4ad38c389a170ccc9af2: FileNotFoundError(2,
↪ 'No such file or directory')
  warnings.warn('resource_tracker: %s: %r' % (name, e))
/Applications/miniconda3/envs/py37cmp-eeg/lib/python3.7/site-packages/joblib/externals/
↪loky/backend/resource_tracker.py:333: UserWarning: resource_tracker: /var/folders/vy/
↪0bw_1jvj54n8lvcgvdrtqb0c0000gn/T/joblib_memmapping_folder_47551_
↪379a43977f814190bfe6f03e92cfdae6_5d99e23deff54a888a0efa2ed85d7b6f: FileNotFoundError(2,
↪ 'No such file or directory')
  warnings.warn('resource_tracker: %s: %r' % (name, e))
/Applications/miniconda3/envs/py37cmp-eeg/lib/python3.7/site-packages/joblib/externals/
↪loky/backend/resource_tracker.py:333: UserWarning: resource_tracker: /var/folders/vy/
↪0bw_1jvj54n8lvcgvdrtqb0c0000gn/T/joblib_memmapping_folder_47551_
↪ad22804ddb624e55b2ee83b946e96936_428a96c1454840a28c73e60ad46a69c8: FileNotFoundError(2,
↪ 'No such file or directory')
  warnings.warn('resource_tracker: %s: %r' % (name, e))
/Applications/miniconda3/envs/py37cmp-eeg/lib/python3.7/site-packages/joblib/externals/
↪loky/backend/resource_tracker.py:333: UserWarning: resource_tracker: /var/folders/vy/
↪0bw_1jvj54n8lvcgvdrtqb0c0000gn/T/joblib_memmapping_folder_47551_
↪379a43977f814190bfe6f03e92cfdae6_d7a66cd336f341f8909b02f329faff21: FileNotFoundError(2,
↪ 'No such file or directory')
  warnings.warn('resource_tracker: %s: %r' % (name, e))
```

A closer look at the EEG pipeline outputs

Let's have a closer look at the outputs that the EEG pipeline produces in the `derivatives/cmp-v3.1.0` derivatives directory.

First of all, connectomemapper works in such a way that the pipeline is first assembled and only afterwards, it is executed. During the assembly stage, input and output variables are connected and CMP3 produces a graph that visualizes this.

[15]:
```
%matplotlib inline
path_to_svg = os.path.join(
    output_dir, __nipype_directory__, participant_label, 'eeg_pipeline','graph.svg'
)
display(SVG(filename=path_to_svg))
```

You can see three blue boxes that represent the different stages of the pipeline flow:

```
* EEG preprocessing stage
* EEG source imaging stage
* EEG connectome stage
```

Each of the stages, again, has an input and and output node, as well as several nodes representing processing steps. Each processing step has its own "interface" which you can find in `cmtklib/interfaces` ("mne" in parentheses indicates that they are defined in the file mne.py).

`eeg_datasource` is the input BIDSDataGrabber node and `eeg_datasinker` is the output DataSinker node. `datasource` takes care of querying and injecting the input files in the different stages of the EEG pipeline. `eeg_sinker` is taking care of collecting, moving, and renaming all the files produced by the different stages to the `derivatives/cmp-v3.1.0` directory.

In the following, we will go over the interfaces and show what output they produce.

EEG preprocessing stage

The preprocessing stage consists of converting EEGLab `.set` EEG files to MNE Epochs in `.fif` format, the format used in the rest of the pipeline by calling, if necessary the following interface:

- `EEGLAB2fif`: Read EEGLab data and converts them to MNE format (.fif file extension).

The information given by the config file regarding this stage is as follows:

```
[...]
    "eeg_preprocessing_stage": {
        "task_label": "faces",
        "eeg_ts_file.extension": "set",
        "eeg_ts_file.toolbox_derivatives_dir": "eeglab-v14.1.1",
        "eeg_ts_file.datatype": "eeg",
        "eeg_ts_file.suffix": "eeg",
        "eeg_ts_file.desc": "preproc",
        "eeg_ts_file.task": "faces",
        "events_file.datatype": "eeg",
        "events_file.suffix": "events",
        "events_file.extension": "tsv",
        "events_file.task": "faces",
        "electrodes_file_fmt": "Cartool",
```

(continues on next page)

(continued from previous page)

```
        "cartool_electrodes_file.toolbox_derivatives_dir": "cartool-v3.80",
        "cartool_electrodes_file.datatype": "eeg",
        "cartool_electrodes_file.suffix": "eeg",
        "cartool_electrodes_file.extension": "xyz",
        "t_min": -0.2,
        "t_max": 0.5
    },
[...]
```

EEGLAB2fif

If your data are not already in MNE format (`.fif` file extension), they have to be read and re-saved. The eeglab2fif interface does this for EEGLAB-format data (`.set` file extension).

The interface produces a file named `sub-01_epo.fif` in the `derivatives/cmp-v3.0.3` folder.

Critically, the saved epochs contain a montage, i.e. the sensor locations which have to be supplied in a file names `sub-01.xyz` inside the subject's EEGLAB derivatives folder (`derivatives/eeglab-v14.1.1/sub-01/eeg/sub-01.xyz`). *Not sure it still applied :)*

[16]:
```
# Let's have a look at the EEG data
with warnings.catch_warnings(): # suppress some irrelevant warnings coming from mne.read_
↪epochs_eeglab()
    warnings.simplefilter("ignore")
    epochs_eeglab = mne.read_epochs_eeglab(
        os.path.join(output_dir, __eeglab_directory__,
                     participant_label, 'eeg',
                     participant_label + f'_task-{task_label}_desc-preproc_eeg.set')
    ) # sub-01_FACES_250HZ_prepd.set

# eeglab2fif removes a baseline and crops the epochs according to parameters start_t and␣
↪end_t in config file
start_t = -0.2
end_t = 0.6
epochs_eeglab.apply_baseline((start_t, 0))
epochs_eeglab.crop(tmin=start_t, tmax=end_t)
evoked_eeglab = epochs_eeglab.average().pick('eeg')

# compare to what eeglab2fif saved
epochs_mne = mne.read_epochs(
    os.path.join(output_dir, __cmp_directory__,
                 participant_label, 'eeg',
                 participant_label + f'_task-{task_label}_epo.fif'))
evoked_mne = epochs_mne.average().pick('eeg')
```

```
Extracting parameters from /Users/sebastientourbier/Documents/GitHub/connectomemapper3/
↪docs/notebooks/ds003505_demo/derivatives/eeglab-v14.1.1/sub-01/eeg/sub-01_task-faces_
↪desc-preproc_eeg.set...
Not setting metadata
Not setting metadata
588 matching events found
No baseline correction applied
```

(continues on next page)

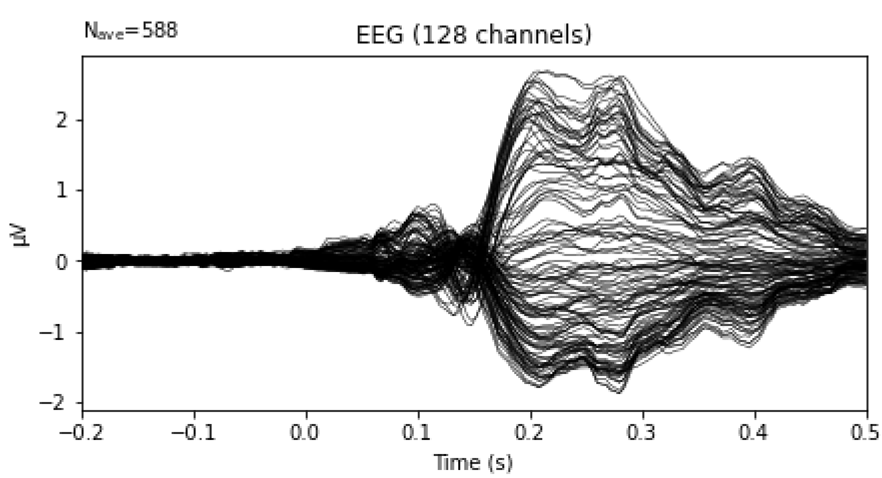
(continued from previous page)

```
0 projection items activated
Ready.
Applying baseline correction (mode: mean)
Reading /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/notebooks/
↪ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/sub-01_task-faces_epo.fif ...
    Read a total of 1 projection items:
        Average EEG reference (1 x 128)  idle
    Found the data of interest:
        t =    -200.00 ...     500.00 ms
        0 CTF compensation matrices available
Not setting metadata
Not setting metadata
588 matching events found
No baseline correction applied
Created an SSP operator (subspace dimension = 1)
1 projection items activated
```

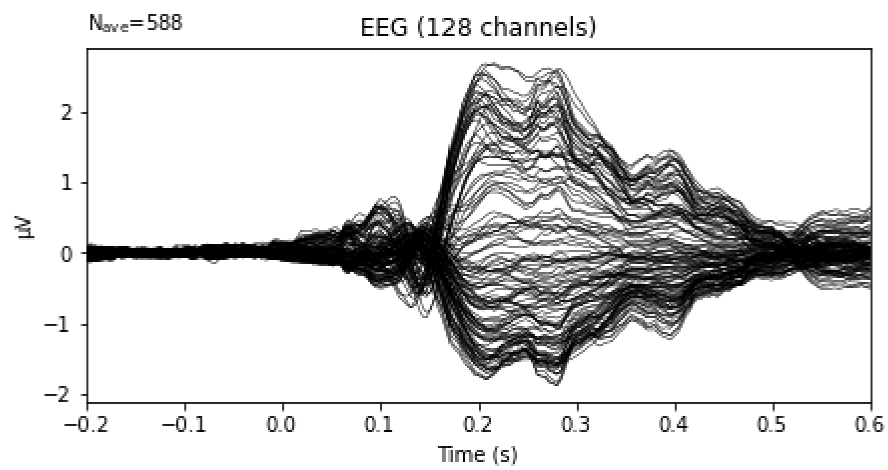
[17]:

```
# plot and convince yourself it's the same
%matplotlib inline
fig = plt.figure()
plt.rcParams['figure.figsize'] = (15, 10)

_ = evoked_mne.plot(time_unit='s')

fig = plt.figure()
plt.rcParams['figure.figsize'] = (15, 10)

_ = evoked_eeglab.plot(time_unit='s')
```

```
<Figure size 432x288 with 0 Axes>
```




```
<Figure size 1080x720 with 0 Axes>
```



EEG source imaging stage

This stage takes your data in fif format from the "Preprocessing Stage", the parcellation, and the previously generated electrode transform file as inputs. With the aim to compute inverse solutions and extract ROI time courses with MNE, its workflow consists of five processing interfaces, :

- `CreateBEM`: Create the boundary element method.
- `CreateSrc`: Create the dipole locations along the surface of the brain.
- `CreateFwd`: Create the forward solution (leadfield) from the BEM and the source space.
- `CreateCov`: Create the noise covariance matrix from the data.
- `MNEInverseSolutionROI`: Create the actual inverse operator and applies it, resulting in ROI-time courses.

The following possible EEG source imaging algorithms can be used for computing the inverse solutions: "sLORETA", "eLORETA", "MNE", and "dSPM". The configuration file of this tutorial is set to use "sLORETA".

The information given by the config file regarding this stage is as follows:

```
[...]
    "eeg_source_imaging_stage": {
        "esi_tool": "MNE",
        "mne_apply_electrode_transform": true,
        "mne_electrode_transform_file.toolbox_derivatives_dir": "cmp-v3.1.0",
        "mne_electrode_transform_file.datatype": "eeg",
        "mne_electrode_transform_file.suffix": "trans",
        "mne_electrode_transform_file.extension": "fif",
        "parcellation_scheme": "Lausanne2018",
        "lausanne2018_parcellation_res": "scale1",
        "mne_esi_method": "sLORETA",
        "mne_esi_method_snr": 3.0
    },
[...]
```

CreateBEM

The BEM (boundary element model) is the head model we use, in our case, it is based on the individual's structural MRI and, again, related freesurfer derivatives. Its creation consists of two steps:

1. The necessary surfaces (brain, inner skull, outer skull, and outer skin) are extracted using `mne.bem.make_watershed_bem()`. The surfaces are saved in the subject's freesurfer-directory in a new folder `bem/watershed`.
2. The model itself is created using `mne.make_bem_model()` and `mne.make_bem_solution()`. In this step, the surfaces and the tissue conductivities between the surfaces are used.

[18]:

```
# Let's visualize the BEM surfaces and source space
src = mne.read_source_spaces(
    os.path.join(
        output_dir,__cmp_directory__,
        participant_label, 'eeg', participant_label + f'_task-{task_label}_src.fif'))
# plot will appear in separate window
%matplotlib qt
# lines are the surfaces, pink dots are the sources (dipoles)
_=mne.viz.plot_bem(
    subject=participant_label,
    subjects_dir=project.freesurfer_subjects_dir,
    brain_surfaces='white',
    src=src,
    orientation='sagittal'
)
```

```
    Reading a source space...
    Computing patch statistics...
    Patch information added...
    Distance information added...
    [done]
    Reading a source space...
    Computing patch statistics...
    Patch information added...
    Distance information added...
    [done]
    2 source spaces read
Using surface: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
↪notebooks/ds003505_demo/derivatives/freesurfer-7.1.1/sub-01/bem/inner_skull.surf
Using surface: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
↪notebooks/ds003505_demo/derivatives/freesurfer-7.1.1/sub-01/bem/outer_skull.surf
Using surface: /Users/sebastientourbier/Documents/GitHub/connectomemapper3/docs/
↪notebooks/ds003505_demo/derivatives/freesurfer-7.1.1/sub-01/bem/outer_skin.surf
```

CreateSrc

MNE is able to create volume- and surface-based source spaces, but in our pipeline, we use surface-based only.

In order to do this, MNE takes advantage of the Freesurfer-created outputs in the `derivatives/freesurfer-7.1.1` derivatives directory.

CreateFwd

MNE first computes a forward solution that describes how electrical currents propagate from the sources created earlier (via createsrc) through the tissues of the head modelled by the BEM (created via createbem) to the electrodes. Thus, the electrode positions have to be known and be aligned to the head model.

[19]:
```
# Let's check the alignment between MRI and electrode positions.
trans = mne.read_trans(
    os.path.join(
        output_dir, __cmp_directory__,
        participant_label, 'eeg', participant_label + '_trans.fif'
    )
)
mne.viz.plot_alignment(
    epochs_mne.info,
    trans=trans,
    subject=participant_label,
    subjects_dir=project.freesurfer_subjects_dir,
    dig=False,
    surfaces=dict(head=0.95),
    coord_frame='mri')
```

```
Using pyvistaqt 3d backend.

Using outer_skin.surf for head surface.
Channel types:: eeg: 128
```

[19]:
```
<mne.viz.backends._pyvista._Figure at 0x7fc713dd9710>
```

CreateCov

MNE uses an estimate of signal to noise ratio in its creation of the inverse solution. For that, it considers the pre-stimulus period of the EEG recordings.

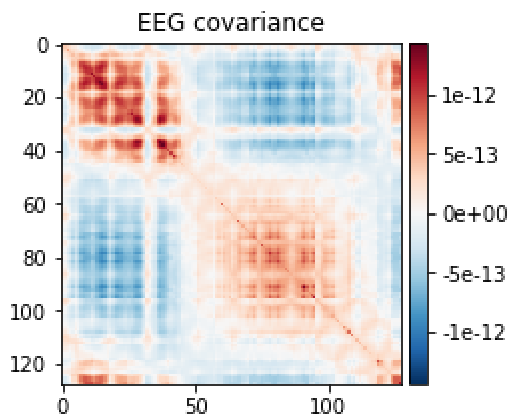
[20]:
```
# Let's have a look at the noise covariance.
%matplotlib inline
noise_cov = mne.read_cov(
    os.path.join(
        output_dir, __cmp_directory__,
        participant_label, 'eeg', participant_label + f'_task-{task_label}_noisecov.fif'
    )
)
fig_cov, fig_spectra = mne.viz.plot_cov(noise_cov, epochs_mne.info)
```

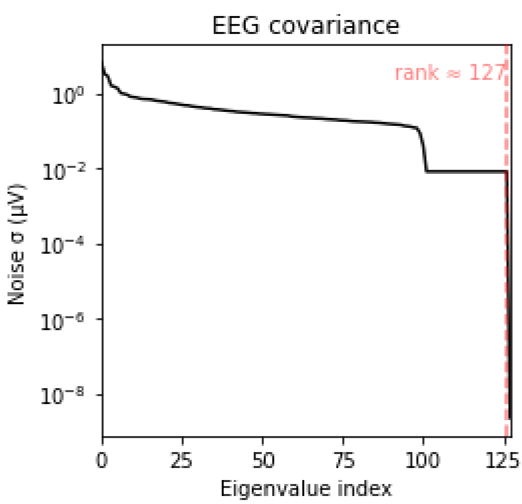
```
    128 x 128 full covariance (kind = 1) found.
    Read a total of 1 projection items:
```

(continues on next page)

(continued from previous page)

```
        Average EEG reference (1 x 128) active
Computing rank from covariance with rank=None
    Using tolerance 1.2e-14 (2.2e-16 eps * 128 dim * 0.43  max singular value)
    Estimated rank (eeg): 127
    EEG: rank 127 computed from 128 data channels with 0 projectors
```





MNEInverseSolutionROI

Now, everything comes together to create the inverse operator, which is then applied to the EEG data to create source time courses. In the last step, the source time courses are converted to ROI-time courses according to the selected parcellation.

The outputs that are necessary for this step to work were created in the previous processing steps, namely:

- the EEG epochs in `.fif` format
- the electrode montage
- the head model
- the source point locations
- the forward operator

- the noise covariance

First, the inverse operator is created using `mne.minimum_norm.make_inverse_operator()`. We use the options `loose=1`, `depth=None`, and `fixed=False` to obtain full 3-dimensional dipoles whose orientation is not fixed or constrained to be (somewhat) orthogonal to surface; and we are not applying any depth weighting. The solution is finally written to a file `sub-01_task-faces_inv.fif` in the same directory as the other outputs (`derivatives/cmp-v3.1.0/sub-01/eeg`).

In a subsequent step in the same interface, this inverse operator is then applied to the epochs (not the evoked time course averaged over trials) using mne.minimum_norm.apply_inverse_epochs.

The final step performed by this interface and by the EEG pipeline is to use mne.extract_label_time_course to create ROI-time courses according to `mne.read_labels_from_annot()`. As given in the config file, we use "lausanne2008" scale 1, which is the Desikan-atlas. The time courses and the ROI-names are stored in `sub-01_task-faces_atlas-L2018_res-scale1_timeseries.pickle` in pickle format.

Let's have a look at the time courses.

[21]:
```
# Load the generated ROI time series file
roi_ts_fname = participant_label + f'_task-{task_label}_atlas-L2018_res-scale1_
↪timeseries.pickle'
roi_ts_file = os.path.join(
    output_dir, __cmp_directory__,
    participant_label, 'eeg', roi_ts_fname
)
with open(roi_ts_file, 'rb') as f:
    rtc_epo = pickle.load(f)
    # For some reason, MNE writes label time courses as lists. convert to numpy array
    rtc_epo['data'] = np.array(rtc_epo['data'])
```

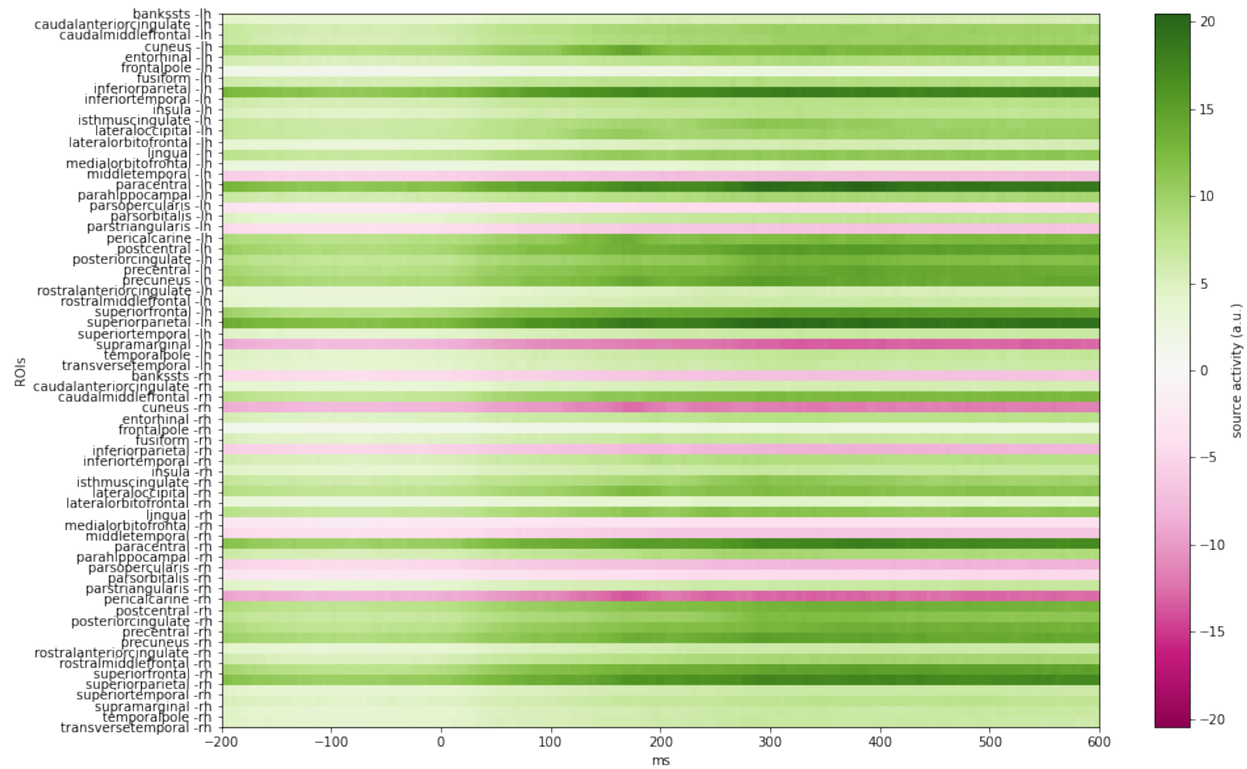
[22]:
```
# Sort labels to make the time courses look nicer
N = len(rtc_epo['labels']) - 2  # two "unknown" regions - do not plot
sorting = list(np.arange(0, N, 2)) + list(np.arange(1, N, 2))  # left and right always␣
↪alternating
# List of ROI names
labels_list_left = [i.name for i in rtc_epo['labels'][0::2] if i.name != 'unknown -lh']
labels_list_right = [i.name for i in rtc_epo['labels'][1::2] if i.name != 'unknown -rh']
labels_list = labels_list_left + labels_list_right
```

[23]:
```
# Plot
%matplotlib inline
to_plot = np.mean(rtc_epo['data'][:, :-2, :], axis=0)
vminmax = np.max(abs(to_plot))
plt.rcParams['figure.figsize'] = (15, 10)
plt.imshow(
    to_plot[sorting, :],
    aspect='auto',
    extent=[-200, 600, 0, 67],
    interpolation='None',
    vmin=-vminmax,
    vmax=vminmax,
    cmap='PiYG'
);
plt.xlabel('ms')
plt.ylabel('ROIs')
```

(continues on next page)

(continued from previous page)

```
cbar = plt.colorbar()
cbar.set_label('source activity (a.u.)')
locs = np.arange(0, N)
_ = plt.yticks(locs, labels_list[-1::-1] )
```



We can see that some of the time courses are "flipped" (have the opposite sign of the others). We will not address this problem here, but this is because of the step where dipole time courses are summarized for each brain region, using PCA. The direction of the resulting vector is not uniquely defined.

This leads us to the last stage of the pipeline, the "Connectome Stage".

EEG connectome stage

This stage aims to map the connectome from the extracted ROI time series and consists of one processing step:

- `MNESpectralConnectivity`: Compute frequency- and time-frequency-domain connectivity measures and save the connectom files in different format.

The information given by the config file regarding this stage is as follows:

```
[...]
    "eeg_connectome_stage": {
        "connectivity_metrics": [
            "coh",
            "cohy",
            "imcoh",
            "plv",
            "ciplv",
```

(continues on next page)

(continued from previous page)

```
            "ppc",
            "pli",
            "wpli",
            "wpli2_debiased"
        ],
        "output_types": [
            "tsv"
            "gPickle",
            "mat",
            "graphml"
        ]
    },
[...]
```

MNESpectralConnectivity

CMP3 uses MNE-Connectivity to compute the functional connectivity matrices. Results can be saved in the same formats (`['tsv', 'gPickle', 'mat', 'graphml']`) as the diffusion MRI and resting-state fMRI pipelines.

Keep in mind that we only plot a single subject's connectivity here, so it is not surprising if you do not see exactly what you would expect.

We can load the matrices in network format, by reading the gpickle files using Networkx:

[24]:
```
# Index the new CMP3 derivatives including the connectome files
# in the BIDSLayout representation
bids_layout.add_derivatives(os.path.join(project.base_directory, "derivatives", "cmp-v3.
↪1.0"))

# Query the generated connectome gpickle file
bids_query = {
    "subject": participant_label.split('-')[-1],  # Keep the label only, e.g. "01"
    "datatype": 'eeg',
    "atlas": 'L2018',
    "res": 'scale1',
    "suffix": 'connectivity',
    "extension": 'gpickle',
    "return_type": 'filename'
}
cmat_file = bids_layout.get(**bids_query)[0]  # BIDSLayout always return a list

# Load wpli2_debiased connectivity matrix from the connectome gpickle file
weight = "wpli2_debiased"
print(f'Load {weight} connectivity matrix from {cmat_file}')
G = nx.read_gpickle(cmat_file)
A_wpli2_debiased = nx.to_numpy_array(G, weight=weight)
A_wpli2_debiased
```

```
Load dataset_description for: /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.1.0
Load wpli2_debiased connectivity matrix from /Users/sebastientourbier/Documents/GitHub/
↪connectomemapper3/docs/notebooks/ds003505_demo/derivatives/cmp-v3.1.0/sub-01/eeg/sub-
↪01_task-faces_atlas-L2018_res-scale1_conndata-network_connectivity.gpickle
```

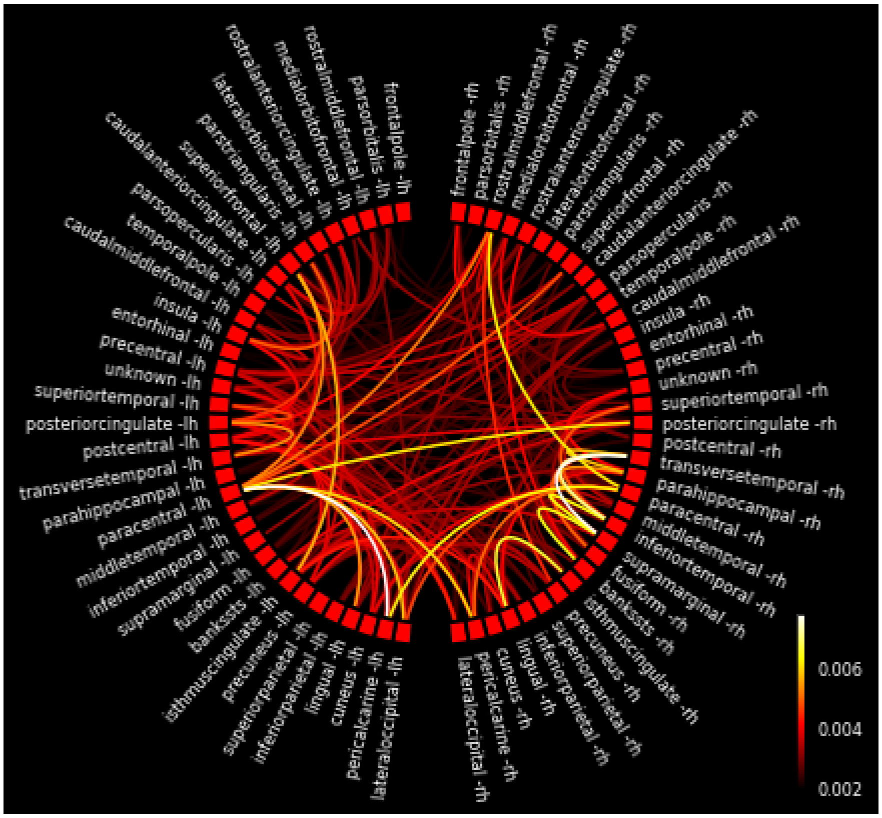
(continues on next page)

(continued from previous page)

[24]:
```
array([[ 0.        ,  0.00300068, -0.00054407, ...,  0.00141492,
        -0.00075917, -0.00026943],
       [ 0.00300068,  0.        ,  0.00172582, ...,  0.00125406,
        -0.00034805, -0.00099333],
       [-0.00054407,  0.00172582,  0.        , ..., -0.0008333 ,
         0.00116659,  0.00129021],
       ...,
       [ 0.00141492,  0.00125406, -0.0008333 , ...,  0.        ,
        -0.00041457,  0.00171886],
       [-0.00075917, -0.00034805,  0.00116659, ..., -0.00041457,
         0.        ,  0.00133674],
       [-0.00026943, -0.00099333,  0.00129021, ...,  0.00171886,
         0.00133674,  0.        ]])
```

Then, we can load and order the name of the labels from the dictionary storing the ROI timeseries results, and visualize the connectivity matrix in a pretty circular layout with MNE-Connectivity `viz.plot_connectivity_circle()` as follows:

[25]:
```
%%time
label_names = [label.name for label in rtc_epo['labels']]

lh_labels = [name for name in label_names if name.endswith('lh')]

# Get the y-location of the label
label_ypos = list()
for name in lh_labels:
    idx = label_names.index(name)
    ypos = np.mean(rtc_epo['labels'][idx].pos[:, 1])
    label_ypos.append(ypos)

# Reorder the labels based on their location
lh_labels = [label for (yp, label) in sorted(zip(label_ypos, lh_labels))]

# For the right hemi
rh_labels = [label[:-2] + 'rh' for label in lh_labels]

# Save the plot order and create a circular layout
node_order = list()
node_order.extend(lh_labels[::-1])  # reverse the order
node_order.extend(rh_labels)

node_angles = mnec.viz.circular_layout(label_names, node_order, start_pos=90,
                              group_boundaries=[0, len(label_names) / 2])

# Plot the graph using node colors from the FreeSurfer parcellation. We only
# show the 300 strongest connections.
# plot will appear in separate window
%matplotlib inline
mnec.viz.plot_connectivity_circle(A_wpli2_debiased, label_names, n_lines=300,
                         node_angles=node_angles, node_colors='r',
                         title='')
```




```
CPU times: user 8.77 s, sys: 47.8 ms, total: 8.82 s
Wall time: 8.41 s
```

[25]:
```
(<Figure size 576x576 with 2 Axes>, <PolarAxesSubplot:>)
```

This concludes the tutorial !

We hope you enjoy it and any feedback or suggestions to improve it are very welcome! Just please open a new issue on GitHub and share your thoughts with us.

[ ]:
```

```

5.10 BSD 3-Clause License

5.11 Changes

5.11.1 Version 3.1.0

Date: MM DD, 2022

This version fully integrates the new pipeline dedicated to EEG modality inside the BIDS App and the GUI.

What's Changed

Updates

- The conda environment files for `cmpbidsappmanager` (`conda/environment.yml` and `conda/environment_macosx.yml`) have been greatly modified (PR #212). This includes the following updates:
  - `python`: from `3.7` to `3.9`
  - `pip`: from `21.3.1` to `22.2`
  - `indexed_gzip`: from `1.6.4` to `1.6.13`
  - `git-annex` (`conda/environment.yml` only): from `8.20211123` to `10.20220724`
  - `qt/pyqt` `5.15.4` installed via `conda-forge`
  - `pyqt5-sip` `12.9.0` (version compatible with `qt/pyqt` `5.15.4`) installed via `conda-forge`

  In addition, the created environment has been renamed `py39cmp-gui` to be consistent with the new python version installed in the environment.
- In all conda environment `*.yml` and `requirements.txt` files, datalad and its container extension have been updated to the following versions (PR #209):
  - `datalad`: from `0.15.4` to `0.17.2` (See Datalad changelog for more details).
  - `datalad-container`: from `1.1.5` to `1.1.6`

New features

- The new pipeline dedicated to EEG modality has been integrated into the BIDS App and `cmpbidsappmanager` (PR #201 and PR #205). EEG pipeline configuration files are passed to the BIDS App or its docker/singularity python wrapper via the option flag `--eeg_pipeline`. A new tab has been added to the configurator window of `cmpbidsappmanager` for the setup and saving of configuration files for the EEG pipeline. A new tab has also been added to the output inspector window of `cmpbidsappmanager` to enable the visual inspection of outputs generated by the EEG pipeline. The EEG configuration file can now be specified in the BIDS App interface window of `cmpbidsappmanager` and the command to run the BIDS has been updated. A new `EEGConnectomeStage` stage has been implemented that builds the connectivity matrices from the extracted

ROI time-series using the function spectral_connectivity_epochs of MNE Connectivity. A new utility script `visualize_eeg_pipeline_outputs.py` has been implemented in the `cmp/cli` module, which is called by the output inspector window of `cmpbidsappmanager`.

- Option to apply or not band-pass filtering in fMRI pipeline. (PR #200)

Code refactoring

- Major refactoring of all the code related to the EEG pipeline (PR #198). This includes:
  - Renaming `EEGLoaderStage` to `EEGPreprocessingStage`,
  - Refactoring inputs/outputs of all interfaces of `cmtklib.eeg`, `cmtklib.interfaces.mne`, and `cmtklib.interfaces.pycartool` modules
  - Refactoring of all inputs, outputs, and config traits of the different stages
  - Modification of (1) `cmp.pipelines.functional.eeg.py` and (2) the tutorial notebook for the EEG pipeline that integrates all previously mentioned changes

Bug fix

- Problems to install and launch cmpbidsappmanager on Ubuntu. (PR #212)
- Fix nibabel to `3.2.2` as the imported functions of `nibabel.trackvis` has been moved since `4.0.0` and caused errors. (PR #XX)
- Fix problem of traits not updated while making the diffusion pipeline config with ACT. (PR #200)

Documentation

- Update/add documentation for the EEG pipeline (PR #208). This includes:
  - Update the BIDS flowchart displayed in `README` and in `docs/index.rst` with the EEG pipeline. The SVG can be found inside the `docs/images/svg` directory.
  - Make appropriate changes to `docs/index.rst` and `README` around the EEG pipeline
  - Show call to `--eeg_pipeline` in `docs/usage.rst`
  - Show how to configure and check outputs of EEG pipeline in `docs/bidsappmanager.rst`
  - Add link to VEPCON dataset as example with EEG in `docs/cmpbids.rst`

Software development life cycle

- Optimization of resources stored in the cache and in the workspace. (PR #201)
- Add tests 10 and 11 that run the EEG pipeline with the MNE and Cartool ESI workflow respectively. (PR #201)

Contributors

- Sebastien Tourbier

More…

Please check the main PR #149 page for more details.

5.11.2 Version 3.0.4

Date: June 15, 2022

This version mainly addresses all points raised by the JOSS review (https://github.com/openjournals/joss-reviews/issues/4248).

What's Changed

Updates

- Nipype has been updated from `1.7.0` to `1.8.0`. (PR #184) See Nipype changelog) for more details.

Bug fix

- Add missing `cmp.stages.eeg` to `setup_pypi.py`. (PR #166)
- Add missing package data for parcellation in `setup_pypi.py`. (PR #182)
- Use HTTPS instead of SSH for datalad clone in notebooks . (PR #181)
- Add missing condition to handle custom BIDS files with session. (PR #183)
- Integrate fix from Napari project for issues with menubar on Mac. (PR #174)
- Use the most recent PyQt5 instead of PySide2 (older) for graphical backend of `cmpbidsappmanager`, which provides a fix to run Qt-based GUI on MacOSX Big Sur. (PR #188)

Documentation

- Correct `conda env create` instruction in the README. (PR #164)
- Refer to contributing guidelines in the README. (PR #167)
- Use `sphinx-copybutton` extension in the docs. (PR #168)
- Add notes about docker image and conda environment size and time to download. (PR #169)

JOSS paper

- Integrate minor wording tweaks by @jsheunis. (PR #162)
- Add higher level summary and rename the old summary to "Overview of Functionalities". (PR #175)

License

- The license has been updated to a pure 3-clause BSD license to comply with JOSS. (PR #163)

Software development life cycle

- Migrate ubuntu 16.04 (now deprecated) to 20.04 on CircleCI. (PR #172)

Contributors

- Sebastien Tourbier
- J.S. (Stephan) Heunis

5.11.3 Version 3.0.3

Date: Feb 18, 2022

This version introduces the new pipeline dedicated to EEG modality with a tutorial, updates Freesurfer to 7.1.1, and adds a new tutorial that shows how to analyze the CMP3 connectomes.

What's Changed

New features

- CMP3 provides a new pipeline `cmp.pipelines.functional.eeg.EEGPipeline` dedicated to EEG modality with a collection of interfaces implemented by the following modules: `cmtklib.eeg`, `cmtklib.interfaces.eeg`, `cmtklib.interfaces.mne`, and `cmtklib.interfaces.pycartool`. See PR #82 for more details.

Updates

- Freesurfer has been updated from `6.1.0` to `7.1.1`. See PR #147 for more details.

Bug fix

- FIX: List of outputs are empty in inspector window of the parcellation and fmri_connectome stages. See PR #145 for more details.
- Correct way GM mask is generated and clean code in cmtklib/parcellation.py.
- Add interface to copy 001.mgz using hardlink.

Documentation

- Add documentation of new classes and functions introduced by the EEG pipeline.
- Add two ipython notebooks in `docs/notebooks` that are integrated directly in the docs with `nbsphinx`:
  - `analysis_tutorial.ipynb`: Show how to interact, analyze, and visualize CMP3 outputs.
  - `EEG_pipeline_tutorial.ipynb`: Show how to use the new API dedicated to the EEG pipeline.

Contributors

- Sebastien Tourbier
- Joan Rue Queralt
- Katharina Glomb
- Mikkel Schoettner

More...

Please check the main PR #146 page for more details.

5.11.4 Version 3.0.2

Date: Jan 31, 2021

This version mostly introduces the capability to estimate carbon footprint of CMP3 execution and fix problem of conflicts during the creation of the conda environment. It incorporates in particular the following changes.

New features

- Allow the estimation of the carbon footprint while using the BIDS App python wrappers and the GUI. Estimations are conducted using codecarbon. All functions supporting this features have been implemented in the new module `cmtklib.carbonfootprint`. See PR #136 for more details.

Code changes

- Creation of `init_subject_derivatives_dirs()` for `AnatomicalPipeline`, `DifusionPipeline`, and `fMRIPipeline` that return the paths to Nipype and CMP derivatives folders of a given subject / session for a given pipeline. This removed all the implicated code from the `process()` method and improve modularity and readability. In the future, the different functions could be merged as there is a lot of code duplication between them.
- `AnatomicalPipeline`, `DiffusionPipeline`, and `fMRIPipeline` workflows are run with the `MultiProc` plugin.

Bug fix

- Major update of the `conda/environment.yml` and `conda/environment_macosx.yml` to correct the problems of conflicts in the previous version, as reported in issue #137. This has resulted in the following package updates:
  - `pip`: 20.1.1 -> 21.3.1
  - `numpy`: 1.19.2 -> 1.21.5
  - `matplotlib`: 3.2.2 -> 3.5.1
  - `traits`: 6.2.0 -> 6.3.2
  - `traitsui`: 7.0.0 -> 7.2.0
  - `graphviz`: 2.40.1 -> 2.50.0
  - `configparser`: 5.0.0 -> 5.2.0
  - `git-annex`: 8.20210127 -> 8.20211123
  - `pyside2`: 5.9.0a1 -> 5.13.2
  - `indexed_gzip`: 1.2.0 -> 1.6.4
  - `cvxpy`: 1.1.7 -> 1.1.18
  - `fsleyes`: 0.33.0 -> 1.3.3
  - `mrtrix3`: 3.0.2 -> 3.0.3
  - `duecredit`: 0.8.0 -> 0.9.1
  - `mne`: 0.20.7 -> 0.24.1
  - `datalad`: 0.14.0 -> 0.15.4
  - `datalad-container`: 1.1.2 -> 1.1.5
  - `statsmodels`: 0.11.1 -> 0.13.1
  - `networkx`: 2.4 -> 2.6.3
  - `pydicom`: 2.0.0 -> 2.2.2

  See commit 483931f for more details.

Documentation

- Add description of carbon footprint estimation feature.
- Improve description on how to use already computed Freesurfer derivatives.

Misc

- Add bootstrap CSS and jquery JS as resources to `cmtklib/data/report/carbonfootprint`. They are used to display the carbon footprint report in the GUI.

- Clean the resources related to parcellation in `cmtklib/data/parcellation` and rename all files and mentions of lausanne2008 to lausanne2018.
- Removed unused `cmtklib.interfaces.camino`, `cmtklib.interfaces.camino2trackvis`, and `cmtklib.interfaces.diffusion` modules
- Specify to `Coverage.py` with `# pragma: no cover` part of the code we know it won't be executed
- Create and use a `coveragerc` file to set the run of `Coverage.py` with `--concurrency=multiprocessing` to be allow to track code inside Nipype interfaces, now managed by multiprocessing.

Code style

- Correct a number of code style issues with class names.

Contributors

- Sebastien Tourbier
- Joan Rue Queralt

More…

Please check the main PR #140 page for more details.

5.11.5 Version 3.0.1

Date: Jan 05, 2021

This version is mostly a bug fix release that allows the python packages of Connectome Mapper 3 to be available on PyPI. It incorporates Pull Request #132 which includes the following changes.

Bug fix

- Rename the project name in `setup.py` and `setup_pypi.py` from `"cmp"` to `"connectomemapper"`. Such a `"cmp"` project name was already existing on PyPI, that caused continuous integration on CircleCI to fail during the last `v3.0.0` release, while uploading the python packages of CMP3 to PyPI.

Code refactoring

- Make `cmp.bidsappmanager.gui.py` more lightweight by splitting the classes defined there in different files. (See Issue #129 for more discussion details)
- Split the `create_workflow()` method of the `RegistrationStage` into the `create_ants_workflow()`, `create_flirt_workflow()`, and `create_bbregister_workflow()`. (See Issue #95 for more discussion details)

Code style

- Correct a number of code style issues with class names

Contributors

- Sebastien Tourbier

Please check the main pull request 132 page for more details.

5.11.6 Version 3.0.0

Date: Dec 24, 2021

This version corresponds to the first official release of Connectome Mapper 3 (CMP3). It incorporates Pull Request #88 (>450 commits) which includes the following changes.

Updates

- traits has been updated from `6.0.0` to `6.2.0`.
- traitsui has been updated from `6.1.3` to `7.0.0`.
- pybids has been updated from `0.10.2` to `0.14.0`.
- nipype has been updated to `1.5.1` to `1.7.0`.
- dipy has been updated from `1.1.0` to `1.3.0`.
- obspy has been updated from `1.2.1` to `1.2.2`.

New features

- CMP3 can take custom segmentation (brain, white-matter, gray-matter and CSF masks, Freesurfer's aparcaseg - used for ACT for PFT) and parcellation files as long as they comply to BIDS Derivatives specifications, by providing the label value for the different entity in the filename. This has led to the creation of the new module `cmtklib.bids.io`, which provides different classes to represent the diversity of custom input BIDS-formatted files. (PR #88)
- CMP3 generates generic label-index mapping `tsv` files along with the parcellation files, in accordance to BIDS derivatives. This has led to the creation of the `CreateBIDSStandardParcellationLabelIndexMappingFile` and `CreateCMPParcellationNodeDescriptionFilesFromBIDSFile` interfaces, which allows us to create the BIDS label-index mapping file from the parcellation node description files employed by CMP3 (that includes `_FreeSurferColorLUT.txt` and `_dseg.graphml`), and vice versa.
- CMP3 provide python wrappers to the Docker and Singularity container images (`connectomemapper3_docker` and `connectomemapper3_singularity`) that will generate and execute the appropriate command to run the BIDS App. (PR #109,

  PR #115, PR #130)

Major changes

- Lausanne2018 parcellation has completely replaced the old Lausanne2008 parcellation. In brief, the new parcellation was introduced to provide (1) symmetry of labels between hemispheres, and (2) a more optimal generation of the volumetric parcellation images, that now are generated at once from `annot` files. This fixes the issue of overwritten labels encountered by in the process of creating the Lausanne2008 parcellation. Any code and data related to Lausanne2008 has been removed. If one still wish to use this old parcellation scheme, one should use CMP3 (`v3.0.0-RC4`).

Output updates

- Directories for the derivatives produced by cmp (`cmp`, `freesurfer`, `nipype`) were renamed to `cmp-`, `freesurfer-`, and `nipype-` to comply with BIDS 1.4.0+. (PR #3 (fork))

Code refactoring

- Creation in `AnatomicalPipeline`, `DiffusionPipeline`, `fMRIPipeline` of `create_datagrabber_node()` and `create_datasinker_node()` methods to reduce the code in `create_workflow()`.
- The `run(command)` function of `cmp.bidsappmanager.core` has been moved to `cmtklib.process`, which is used by the python wrappers in `cmp.cli`.

Pipeline Improvements

- Better handle of existing Freesurfer outputs. In this case, CMP3 does not re-create the `mri/orig/001.mgz` and connect the reconall interface anymore.
- Creation of 5TT, gray-matter / white-matter interface, and partial volume maps images are performed in the preprocessing stage of the diffusion pipeline only if necessary

Code Style

- Clean code and remove a number of commented lines that are now obsolete. Code related to the connection of nodes in the Nipype `Workflow` adopts a specific format and are protected from being reformatted by BLACK with the `# fmt: off` and `# fmt: on` tags.

Documentation

- Add instructions to use custom segmentation and parcellation files as inputs.
- Add description in contributing page of format for code related to the connection of the nodes in a Nipype `Workflow`.
- Add instructions to use the python wrappers for running the BIDS App. (PR #115)
- Add notification about the removal of the old Lausanne2008 parcellation, and remove any other mentions in the documentation.

Software container

- Define multiple build stages in Dockerfile, which can be run in parallel at build with BUILDKIT. (PR #88)

Software development life cycle

- Update the list of outputs of circleci tests with the new names of directories produced by cmp in `output_dir/`.
- Following major changes in the pricing plans of CircleCI but also to improve its readability, `circleci/config.yml` has been dramatically refactored, including: * Use BUILDKIT in docker build to take advantage of the multi-stage build * Reordering and modularization of the tests:
  - tests 01-02 (Docker): anatomical pipeline for each parcellation scheme
  - tests 03-06 (Docker): diffusion pipeline for dipy/mrtrix deterministic/probabilistic tractography
  - tests 07-08 (Docker): fMRI pipeline for FLIRT and BBRegistration registrations
  - test 09 (Singularity): anatomical pipeline for Lausanne2018 scheme
  - Creation of commands for steps that are shared between jobs to reduce code duplication

  (PR #88)

Contributors

- Sebastien Tourbier
- Anil Tuncel
- Jakub Jancovic
- Jonathan Wirsich

Please check the main pull request 88 page for more details.

5.11.7 Version 3.0.0-RC4

Date: March 07, 2021

This version corresponds to the fourth and final release candidate of Connectome Mapper 3 (CMP3). It incorporates the relatively large Pull Request #74 (~270 commits) which includes the following changes such that it marks the end of the release candidate phase.

New features

- CMP3 pipeline configuration files adopt JSON as new format. (PR #76)
- CMP3 is compatible with PyPI for installation. (PR #78)
- BIDS convention naming of data derived from parcellation atlas adopt now the new BIDS entity `atlas-<atlas_label>` to distinguish data derived from different parcellation atlases. The use of the entity `desc-<scale_label>` to distinguish between parcellation scale has been replaced by the use of the entity `res-<scale_label>`. (PR #79)

Updates

- Content of `dataset_description.json` for each derivatives folder has been updated to conform to BIDS version 1.4.0. (PR #79)

Code refactoring

- Major refactoring of the *cmtklib.config* module with the addition and replacement of a number of new methods to handle JSON configuration files. (See full diff on GitHub) Configuration files in the old INI format can be converted automatically with the help of the two new methods `check_configuration_format()` and `convert_config_ini_2_json()` to detect if configuration files are in the INI format and to make the conversion. (PR #76)
- Major changes to make *cmp* and `cmpbidsappmanager` compatible with the Python Package Index (`pip`) for package distribution and installation. This includes the merge of `setup.py` and `setup_gui.py`, which have been merged into one `setup.py` and a major refactoring to make `pip` happy, as well as the creation of a new `cmp.cli` module, migration to `cmp.cli` module and refactoring of the scripts `connectomemapper3`, `showmatrix_gpickle`, and `cmpbidsappmanager` with correction of code style issues and addition of missing docstrings. (PR #78)

Improvements

- Clean parameters to be saved in configuration files with the new API. (PR #74)
- Clean output printed by the cmpbidsappmanager Graphical User Interface. (PR #74)
- Add in *cmtklib.config* the three new functions `print_error`, `print_blue`, and `print_warning` to use different colors to differentiate general info (default color), error (red), command or action (blue), and highlight or warning (yellow). (PR #74)
- Clean code and remove a number of commented lines that are now obsolete. (PR #74, PR #79)

Documentation

- Review usage and add a note regarding the adoption of the new JSON format for configuration files. (PR #76)
- Update tutorial on using CMP3 and Datalad for collaboration. (PR #77)
- Update installation instruction of `cmpbidsappmanager` using `pip install .`. (PR #78)
- Update list of outputs following the new BIDS derivatives naming convention introduced. (PR #79)

Bug fixes

- Correct attributes related to the diffusion imaging model type `multishell`. (PR #74)

- Review code in `cmtklib/connectome.py` for saving functional connectome files in GRAPHML format. (PR #74)

Software Updates

- Update version of datalad and dependencies (PR #77):
  - `datalad[full]==0.13.0` to `datalad[full]==0.14.0`.
  - `datalad-container==0.3.1` to `datalad-container==1.1.2`.
  - `datalad_neuroimaging==0.2.0` to `datalad-neuroimaging==0.3.1`.
  - `git-annex=8.20200617` to `git-annex=8.20210127`.
  - `datalad-revolution` was removed.

Software development life cycle

- Improve code coverage by calling the methods `check_stages_execution()` and `fill_stages_outputs()` on each pipeline when executed with coverage. (PR #75)
- Improve code coverage by saving in test-01 structural connectome files in MAT and GRAPHML format. (PR #74)
- Improve code coverage by saving in test-07 functional connectome files in GRAPHML format. (PR #74)
- Update the list of outputs for all tests. (PR #74)
- Add `test-python-install` job that test the build and installation of *cmp* and `cmpbidsappmanager` packages compatible with `pip`. (PR #78)

Please check the main pull request 74 page for more details.

5.11.8 Version 3.0.0-RC3

Date: February 05, 2021

This version corresponds to the third release candidate of Connectome Mapper 3. In particular, it integrates Pull Request #62 which includes:

Updates

- MRtrix3 has been updated from `3.0_RC3_latest` to `3.0.2`.
- Numpy has been updated from `1.18.5` to `1.19.2`.
- Nipype has been updated to `1.5.0` to `1.5.1`.
- Dipy has been updated from `1.0.0` to `1.3.0`.
- CVXPY has been updated from `1.1.5` to `1.1.7`.

Documentation

- Update outdated screenshots for GUI documentation page at readthedocs reported at CMTK user-group.
- Correction of multiple typos.

Bug fixes

- Update code for Dipy tracking with DTI model following major changes in Dipy 1.0 (Fix reported issue #54).
- Update to Dipy 1.3.0 has removed the deprecated warnings related to CVXPY when using MAP_MRI (#63)
- Do not set anymore `OMP_NUM_THREADS` at execution due to allocation errors raised when using numpy function dot in Dipy.

Software development life cycle

- Add `Test 08` that runs anatomical and fMRI pipelines with: Lausanne2018 parcellation, FSL FLIRT co-registration, all nuisance regression, linear detrending and scrubbing
- Add `Test 09` that runs anatomical and dMRI pipelines with: Lausanne2018 parcellation, FSL FLIRT, Dipy SHORE, MRtrix SD_Stream tracking, MRtrix SIFT tractogram filtering
- Remove `deploy_singularity_latest` from the workflow for the sake of space on Sylabs.io.

Please check the main pull request 62 page for more details.

5.11.9 Version 3.0.0-RC2-patch1

Date: February 4, 2021

This version fixes bugs in the second release candidate of Connectome Mapper 3 (v3.0.0-RC2). In particular, it includes:

Bug fixes

- Fix the error to save connectome in GraphML format reported in #65 and (Pull Request #66).

Software development life cycle

- Remove publication of the Singularity image to sylabs.io when the master branch is updated for the sake of space (11GB limit)

Commits

- CI: remove publication of latest tag image on sylabs.io for space (2 days ago) - commit c765f79
- Merge pull request #66 from connectomicslab/v3.0.0-RC2-hotfix1 (3 days ago) - commit 0a2603e
- FIX: update g2.node to g2.nodes when saving connectomes as graphml (fix #65) (6 days ago) - commit d629eef
- FIX: enabled/disabled gray-out button "Run BIDS App" with Qt Style sheet [skip ci] (3 weeks ago) - commit 10e78d9
- MAINT: removed commented lines in cmpbidsappmanager/gui.py [skip ci] (3 weeks ago) - commit 4cc11e7
- FIX: check availability of modalities in the BIDS App manager window [skip ci] (3 weeks ago) - commit 80fbee2
- MAINT: update copyright year [skip ci] (3 weeks ago) - commit f7d0ffb
- CI: delete previous container with latest TAG on sylabs.io [skip ci] (4 weeks ago) - commit 15c9b18
- DOC: update tag to latest in runonhpc.rst [skip ci] (4 weeks ago) - commit 3165bcc
- CI: comment lines related to version for singularity push (4 weeks ago) - commit 3952d46

5.11.10 Version 3.0.0-RC2

Date: December 24, 2020

This version corresponds to the second release candidate of Connectome Mapper 3. In particular, it integrates Pull Request #45 which includes:

New feature

- Add SIFT2 tractogram filtering (requested in #48, PR #52).
- Add a tracker to support us seeking for new funding. User is still free to opt-out and disable it with the new option flag `--notrack`.

- Add options suggested by Theaud G et al. (2020) to better control factors having impacts on reproducibility. It includes:
  - Set the number of ITK threads used by ANTs for registration (option flag `--ants_number_of_threads`).
  - Set the seed of the random number generator used by ANTs for registration (option flag `--ants_random_seed`).
  - Set the seed of the random number generator used by MRtrix for tractography seeding and track propagation (option flag `--mrtrix_random_seed`).
- Full support of Singularity (see Software development life cycle).

Code refactoring

- A number of classes describing interfaces to `fsl` and `mrtrix3` have been moved from `cmtklib/interfaces/util.py` to `cmtklib/interfaces/fsl.py` and `cmtklib/interfaces/mrtrix3.py`.
- Capitalize the first letter of a number of class names.
- Lowercase a number of variable names in `cmtklib/parcellation.py`.

Graphical User Interface

- Improve display of qpushbuttons with images in the GUI (PR #52).
- Make the window to control BIDS App execution scrollable.
- Allow to specify a custom output directory.
- Tune new options in the window to control BIDS App multi-threading (OpenMP and ANTs) and random number generators (ANTs and MRtrix).

Documentation

- Full code documentation with *numpydoc*-style docstrings.
- API documentation page at readthedocs.

Bug fixes

- Fix the error reported in #17 if it is still occuring.
- Review statements for creating contents of BIDS App entrypoint scripts to fix issue with Singularity converted images reported in #47.
- Install `dc` package inside the BIDS App to fix the issue with FSL BET reported in #50.
- Install `libopenblas` package inside the BIDS App to fix the issue with FSL EDDY_OPENMP reported in #49.

Software development life cycle

- Add a new job `test_docker_fmri` that test the fMRI pipeline.
- Add `build_singularity`, `test_singularity_parcellation`, `deploy_singularity_latest`, and `deploy_singularity_release` jobs to build, test and deploy the Singularity image in CircleCI (PR #56).

Please check the main pull request 45 page for more details.

5.11.11 Version 3.0.0-RC1

Date: August 03, 2020

This version corresponds to the first release candidate of Connectome Mapper 3. In particular, it integrates Pull Request #40 where the last major changes prior to its official release have been made, which includes in particular:

Migration to Python 3

- Fixes automatically with `2to3` and manually a number of Python 2 statements invalid in python 3 including the print() function
- Correct automatically PEP8 code style issues with autopep8
- Correct manually a number of code stly issues reported by Codacy (bandits/pylints/flake8)
- Major dependency upgrades including:
  - `dipy 0.15 -> 1.0` and related code changes in `cmtklib/interfaces/dipy` (Check here for more details about Dipy 1.0 changes)

> **Warning:** Interface for tractography based on Dipy DTI model and EuDX tractography, which has been drastically changed in Dipy 1.0, has not been updated yet, It will be part of the next release candidate.

  - `nipype 1.1.8 -> 1.5.0`
  - `pybids 0.9.5 -> 0.10.2`
  - `pydicom 1.4.2 -> 2.0.0`
  - `networkX 2.2 -> 2.4`
  - `statsmodels 0.9.0 -> 0.11.1`
  - `obspy 1.1.1 -> 1.2.1`
  - `traits 5.1 -> 6.0.0`
  - `traitsui 6.0.0 -> 6.1.3`
  - `numpy 1.15.4 -> 1.18.5`
  - `matplotlib 1.1.8 -> 1.5.0`
  - `fsleyes 0.27.3 -> 0.33.0`
  - `mne 0.17.1 -> 0.20.7`
  - `sphinx 1.8.5 -> 3.1.1`
  - `sphinx_rtd_theme 0.4.3 -> 0.5.0`
  - `recommonmark 0.5.0 -> 0.6.0`

New feature

- Option to run Freesurfer recon-all in parallel and to specify the number of threads used by not only Freesurfer but also all softwares relying on OpenMP for multi-threading. This can be achieved by running the BIDS App with the new option flag `--number_of_threads`.

Changes in BIDS derivatives

- Renamed connectivity graph files to better conform to the BIDS extension proposal on connectivity data schema. They are now saved by default in a TSV file as a list of edges.

Code refactoring

- Functions to save and load pipeline configuration files have been moved to `cmtklib/config.py`

Bug fixes

- Major changes in how inspection of stage/pipeline outputs with the graphical user interface (cmpbidsappmanager) which was not working anymore after migration to Python3
- Fixes to compute the structural connectivity matrices following migration to python 3
- Fixes to computes ROI volumetry for Lausanne2008 and NativeFreesurfer parcellation schemes
- Add missing renaming of the ROI volumetry file for the NativeFreesurfer parcellation scheme following BIDS
- Create the mask used for computing peaks from the Dipy CSD model when performing Particle Filtering Tractography (development still on-going)
- Add missing renaming of Dipy tensor-related maps (AD, RD, MD) following BIDS
- Remove all references to use Custom segmentation / parcellation / diffusion FOD image / tractogram, inherited from CMP2 but not anymore functional following the adoption of BIDS standard inside CMP3.

Software development life cycle

- Use Codacy to support code reviews and monitor code quality over time.
- Use coveragepy in CircleCI during regression tests of the BIDS app and create code coverage reports published on our Codacy project page.
- **Add new regression tests in CircleCI to improve code coverage:**
  - Test 01: Lausanne2018 (full) parcellation + Dipy SHORE + Mrtrix3 SD_STREAM tractography
  - Test 02: Lausanne2018 (full) parcellation + Dipy SHORE + Mrtrix3 ACT iFOV2 tractography
  - Test 03: Lausanne2018 (full) parcellation + Dipy SHORE + Dipy deterministic tractography
  - Test 04: Lausanne2018 (full) parcellation + Dipy SHORE + Dipy Particle Filtering tractography
  - Test 05: Native Freesurfer (Desikan-Killiany) parcellation
  - Test 06: Lausanne2008 parcellation (as implemented in CMP2)
- Moved pipeline configurations for regression tests in CircleCI from `config/` to `.circle/tests/configuration_files`
- Moved lists of expected regression test outputs in CircleCI from `.circle/` to `.circle/tests/expected_outputs`

Please check the pull request 40 page for more details.

5.11.12 Version 3.0.0-beta-RC2

Date: June 02, 2020

This version integrates Pull Request #33 which corresponds to the last beta release that still relies on Python 2.7. It includes in particular:

Upgrade

- Uses `fsleyes` instead of `fslview` (now deprecated), which now included in the conda environment of the GUI (`py27cmp-gui`).

New feature

- Computes of ROI volumetry stored in `<output_dir>/sub-<label>(/ses<label>)/anat` folder, recognized by their `_stats.tsv` file name suffix.

Improved replicability

- Sets the `MATRIX_RNG_SEED` environment variable (used by MRtrix) and seed for the numpy random number generator (`numpy.random.seed()`)

Bug fixes

- Fixes the output inspector window of the cmpbidsappmanager (GUI) that fails to find existing outputs, after adoption of /bids_dir and /output_dir in the bidsapp docker image.
- Fixes the way to get the list of networkx edge attributes in `inspect_outputs()` of `ConnectomeStage` for the output inspector window of the cmpbidsappmanager (GUI)
- Added missing package dependencies (`fury` and `vtk`) that fixes dipy_CSD execution error when trying to import module actor from dipy.viz to save the results in a png
- Fixes a number of unresolved references identified by pycharm code inspection tool

Code refactoring

- Interfaces for fMRI processing were moved to `cmtklib/functionalMRI.py`.
- Interface for fMRI connectome creation (`rsfmri_conmat`) moved to `cmtklib/connectome.py`

Please check the pull request 33 page for change details.

5.11.13 Version 3.0.0-beta-RC1

Date: March 26, 2020

This version integrates Pull Request #28 which includes in summary:

- A major revision of continuous integration testing and deployment with CircleCI which closes Issue 14 integrates an in-house dataset published and available on Zenodo @ https://doi.org/10.5281/zenodo.3708962.
- Multiple bug fixes and enhancements incl. close Issue 30 , update mrtrix3 to RC3 version, bids-app run command generated by the GUI, location of the configuration and log files to be more BIDS compliant.
- Change in tagging beta version which otherwise might not be meaningfull in accordance with the release date (especially when the expected date is delayed due to unexpected errors that might take longer to be fixed than expected).

Please check the pull request 28 page for a full list of changes.

5.11.14 Version 3.0.0-beta-20200227

Date: February 27, 2020

This version addresses multiple issues to make successful conversion and run of the CMP3 BIDS App on HPC (Clusters) using Singularity.

- Revised the build of the master and BIDS App images:
  - Install locales and set `$LC_ALL` and `$LANG` to make freesurfer hippocampal subfields and brainstem segmentation (matlab-based) modules working when run in the converted SIngularity image
  - BIDS input and output directories inside the BIDS App container are no longer the `/tmp` and `/tmp/derivatives` folders but `/bids_dir` and `/output_dir`. .. warning:: this might affect the use of Datalad container (To be confirmed.)

  - Fix the branch of mrtrix3 to check out
  - Updated metadata
- Fix the configuration of CircleCI to not use Docker layer cache feature anymore as this feature is not included anymore in the free plan for open source projects.
- Improved documentation where the latest version should be dynamically generated everywhere it should appear.

5.11.15 Version 3.0.0-beta-20200206

Date: February 06, 2020

- Implementation of an in-house Nipype interface to AFNI 3DBandPass which can handle to check output as ..++orig.BRIK or as ..tlrc.BRIK (The later can occur with HCP preprocessed fmri data)

5.11.16 Version 3.0.0-beta-20200124

Date: January 24, 2020

- Updated multi-scale parcellation with a new symmetric version:
  1. The right hemisphere labels were projected in the left hemisphere to create a symmetric version of the multiscale cortical parcellation proposed by Cammoun2012.
  2. For scale 1, the boundaries of the projected regions over the left hemisphere were matched to the boundaries of the original parcellation for the left hemisphere.
  3. This transformation was applied for the rest of the scales.
- Updated documentation with list of changes

5.12 Citing

Important:

- If your are using the Connectome Mapper 3 in your work, please acknowledge this software with the following two entries:

  1. Tourbier S, Aleman-Gomez Y, Mullier E, Griffa A, Wirsich J, Tuncel MA, Jancovic J, Bach Cuadra M, Hagmann P, (2022). Connectome Mapper 3: A Flexible and Open-Source Pipeline Software for Multiscale Multimodal Human Connectome Mapping. Journal of Open Source Software, 7(74), 4248, https://doi.org/10.21105/joss.04248

```
@article{TourbierJOSS2022,
    doi = {10.21105/joss.04248},
    url = {https://doi.org/10.21105/joss.04248},
    year = {2022},
    publisher = {{The Open Journal}},
    volume = {7},
    number = {74},
    pages = {4248},
    author = {Tourbier, Sebastien and
              Rue Queralt, Joan and
```

(continues on next page)

(continued from previous page)

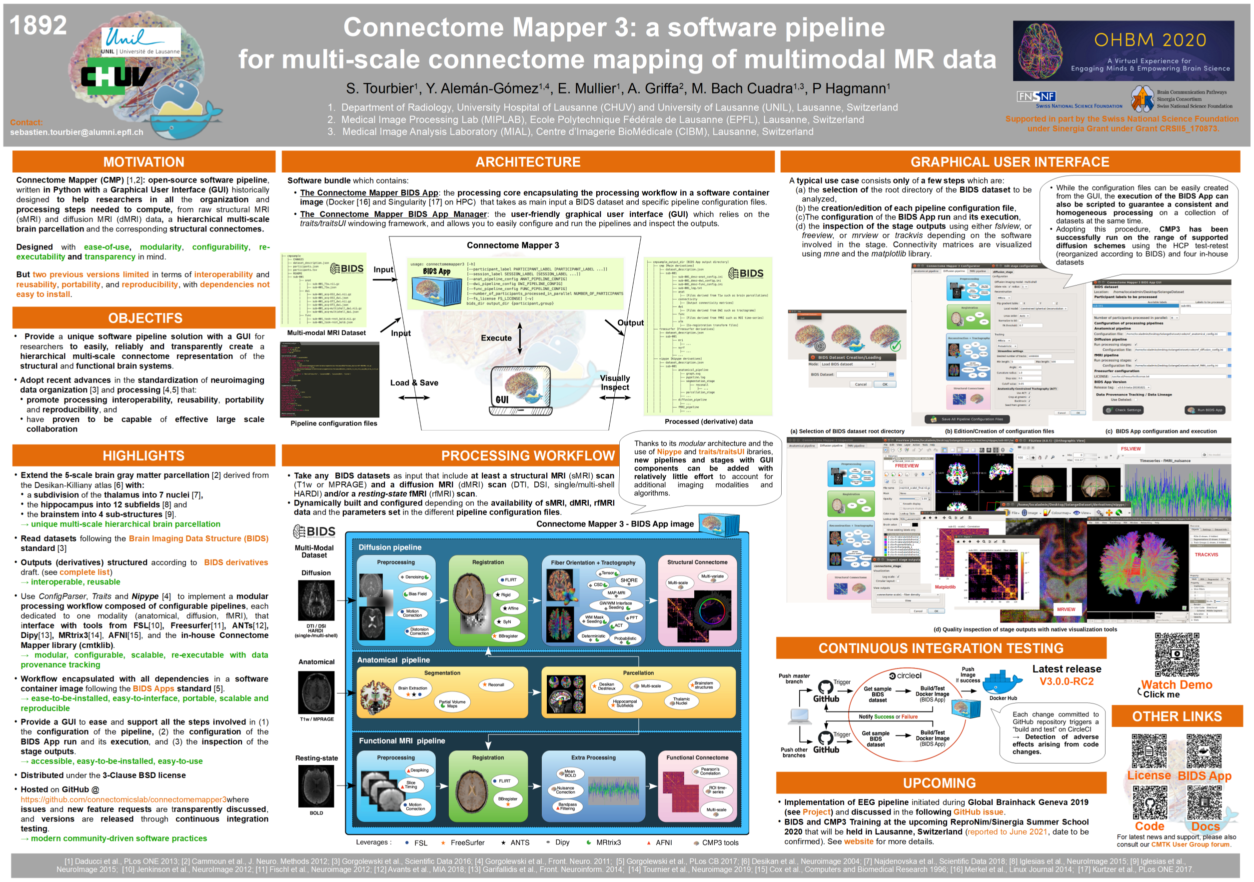
```
                    Glomb, Katharina and
                    Aleman-Gomez, Yasser and
                    Mullier, Emeline and
                    Griffa, Alessandra and
                    Schöttner, Mikkel and
                    Wirsich, Jonathan and
                    Tuncel, Anil and
                    Jancovic, Jakub and
                    Bach Cuadra, Meritxell and
                    Hagmann, Patric},
    title = {{Connectome Mapper 3: A Flexible and Open-Source
                    Pipeline Software for Multiscale Multimodal Human
                    Connectome Mapping},
    journal = {{Journal of Open Source Software}}
}
```

2. Tourbier S, Aleman-Gomez Y, Mullier E, Griffa A, Wirsich J, Tuncel MA, Jancovic J, Bach Cuadra M, Hagmann P. (2022). Connectome Mapper 3: A Flexible and Open-Source Pipeline Software for Multiscale Multimodal Human Connectome Mapping (v3.1.0). Zenodo. http://doi.org/10.5281/zenodo.6645256.

```
@software{TourbierZenodo6645256,
    author = {Tourbier, Sebastien and
                    Rue Queralt, Joan and
                    Glomb, Katharina and
                    Aleman-Gomez, Yasser and
                    Mullier, Emeline and
                    Griffa, Alessandra and
                    Schöttner, Mikkel and
                    Wirsich, Jonathan and
                    Tuncel, Anil and
                    Jancovic, Jakub and
                    Bach Cuadra, Meritxell and
                    Hagmann, Patric},
    title = {{Connectome Mapper 3: A Flexible and Open-Source
                    Pipeline Software for Multiscale Multimodal Human
                    Connectome Mapping}},
    month = jun,
    year = 2022,
    publisher = {Zenodo},
    version = {v3.0.4},
    doi = {10.5281/zenodo.6645256},
    url = {https://doi.org/10.5281/zenodo.6645256}
}
```

5.12.1 Poster

- Organization for Human Brain Mapping 2020 (Abstract; Poster)



5.13 Contributors

Thanks goes to these wonderful people (emoji key):

Thanks also goes to all these wonderful people that contributed to the two first versions of Connectome Mapper:

- Collaborators from Signal Processing Laboratory (LTS5), EPFL, Lausanne:
  - Jean-Philippe Thiran
  - Leila Cammoun
  - Adrien Birbaumer (abirba)
  - Alessandro Daducci (daducci)
  - Stephan Gerhard (unidesigner)
  - Christophe Chênes (Cwis)
  - Oscar Esteban (oesteban)
  - David Romascano (davidrs06)
  - Alia Lemkaddem (allem)

- Xavier Gigandet
- Collaborators from Children's Hospital, Boston:
  - Ellen Grant
  - Daniel Ginsburg (danginsburg)
  - Rudolph Pienaar (rudolphpienaar)
  - Nicolas Rannou (NicolasRannou)

This project follows the all-contributors specification. Contributions of any kind are welcome!

See *contributing page* for more details about how to join us!

5.14 Contributing to Connectome Mapper 3

Contents

5.14.1 Philosophy

The development philosophy for this new version of the Connectome Mapper is to:

I. Enhance interoperability by working with datasets structured following the Brain Imaging Data Structure structured dataset.

II. Keep the code of the processing as much as possible outside of the actual main Connectome Mapper code, through the use and extension of existing Nipype interfaces and an external library (dubbed cmtklib).

III. Separate the code of the graphical interface and the actual main Connectomer Mapper code through inheritance of the classes of the actual main stages and pipelines.

IV. Enhance portability by freezing the computing environment with all software dependencies installed, through the adoption of the BIDS App framework relying on light software container technologies.

V. Adopt best modern open-source software practices that includes to continuously test the build and execution of the BIDS App with code coverage and to follow the PEP8 and PEP257 conventions for python code and docstring style conventions. The use of an integrated development environment such as PyCharm or SublimeText with a python linter (code style checker) is strongly recommended.

VI. Follow the all contributors specification to acknowledge any kind of contribution.

This means that contributions in many different ways (discussed in the following subsections) are welcome and will be properly acknowledged! If you have contributed to `CMP3` and are not listed as contributor, please add yourself and make a pull request.

This also means that further development, typically additions of other tools and configuration options should go in this direction.

5.14.2 Contribution Types

Report Bugs

Report bugs at https://github.com/connectomicslab/connectomemapper3/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” and “help wanted” is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with “enhancement” and “help wanted” is open to whoever wants to implement it.

Possible enhancements are probably to be included in the following list:

I. Adding of a configuration option to an existing stage

II. Adding a new interface to *`cmtklib`*

III. Adding of a new stage

IV. Adding of a new pipeline

The adding of newer configuration options to existing stages should be self-understandable. If the addition is large enough to be considered a “sub-module” of an existing stage, see the Diffusion stage example.

Adding a new stage implies the addition of the stage folder to the `cmp/stages` and `cmp/bidsappmanager/stages` directory and according modification of the parent pipeline along with insertion of a new image in `cmp/bidsappmanager/stages`. Copy-paste of existing stage (such as segmentation stage) is recommended. Note that CMP3 adopts a specific style for code dedicated to the connection of stages and interfaces, which is as follows:

```
[...]
# fmt: off
anat_flow.connect(
    [
        (seg_flow, parc_flow, [("outputnode.subjects_dir", "inputnode.subjects_
↪dir"),
                               ("outputnode.subject_id", "inputnode.subject_id
↪")]),
        (seg_flow, anat_outputnode, [("outputnode.subjects_dir", "subjects_dir
↪"),
                                     ("outputnode.subject_id", "subject_id")]),
        [...]
    ]
)
# fmt: on
[...]
```

The `# fmt: off` and `# fmt: on` flags protect the lines to be reformatted by BLACK.

Adding a new pipeline implies the creation of a new pipeline script and folder in the `cmp/pipelines` and `cmp/bidsappmanager/pipelines` directories Again copy-pasting an existing pipeline is the better idea here. Modification of `cmp/project.py` and `cmp/bidsappmanager/project.py` file is also needed.

Each new module, class or function should be properly documented with a docstring in accordance to the Numpy docstring style.

Write Documentation

`CMP3` could always use more documentation, whether as part of the official CMP3 docs, in docstrings, or even on the web in blog posts, articles, and such.

When you commit changes related to the documentation, please always insert at then end of your message `[skip ci]` to not perform continuous integration of the whole project with CircleCI.

Submit Feedback

The best way to send feedback is to create an issue at https://github.com/connectomicslab/connectomemapper3/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.14.3 Get Started!

Ready to contribute? Here's how to set up `Connectome Mapper 3` for local development.

1. Fork the `connectomemapper3` repo on GitHub.
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/connectomemapper3.git
cd connectomemapper3
```

3. Create a branch for local development:

```
git checkout -b name-of-your-bugfix-or-feature
```

4. Now you can make your changes locally. If you add a new node in a pipeline or a completely new pipeline, we encourage you to rebuild the BIDS App Docker image (See *BIDS App build instructions*).

Note: Please keep your commit the most specific to a change it describes. It is highly advice to track un-staged files with `git status`, add a file involved in the change to the stage one by one with `git add <file>`. The use of `git add .` is highly discouraged. When all the files for a given change are staged, commit the files with a brief message using `git commit -m "[COMMIT_TYPE]: Your detailed description of the change."` that describes your change and where `[COMMIT_TYPE]` can be `[FIX]` for a bug fix, `[ENH]` for a new feature, `[MAINT]` for code maintenance and typo fix, `[DOC]` for documentation, `[CI]` for continuous integration testing, `[UPD]` for dependency update, `[MISC]` for miscellaneous.

---

5. When you're done making changes, push your branch to GitHub:

```
git push origin name-of-your-bugfix-or-feature
```

6. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. If the pull request adds functionality, the docs and tests should be updated (See *documentation build instructions*).
2. Python code and docstring should comply with PEP8 and PEP257 standards.
3. The pull request should pass all tests on GitHub.

How to build the BIDS App locally

1. Go to the clone directory of your fork and run the script `build_bidsapp.sh`

```
cd connectomemapper3
sh scripts/build_bidsapp.sh
```

Note: Tag of the version of the image is extracted from `cmp/info.py`. You might want to change the version in this file to not overwrite an other existing image with the same version.

How to build the documentation locally

To generate the documentation:

1. Install the CMP3 conda environment `py39cmp-gui`:

```
$ cd connectomemapper3
$ conda env create -f environment.yml
```

2. Activate CMP3 conda environment `py39cmp-gui`:

```
$ conda activate py39cmp-gui
```

3. Install all dependencies such as sphinx and its extensions, required for the build:

```
(py39cmp-gui)$ pip install -r docs/requirements.txt
```

4. Install `connectomemapper3`:

```
(py39cmp-gui)$ pip install .
```

5. Run the script `scripts/build_docs.sh` to generate the HTML documentation in `docs/_build/html`:

```
(py39cmp-gui)$ sh scripts/build_docs.sh
```

Note: Make sure to have (1) activated the conda environment `py39cmp-gui` and (2) reinstalled `connectomemapper3` with `pip` before running `build_docs.sh`.

Authors Sebastien Tourbier, Adrien Birbaumer

Version Revision: 2

Acknowledgments

We thanks the authors of these great contributing guidelines, from which part of this document has been inspired and adapted.

5.15 Support, Bugs and New Feature Requests

If you need any support or have any questions, you can post to the CMTK-users group.

All bugs, concerns and enhancement requests for this software are managed on GitHub and can be submitted at https://github.com/connectomicslab/connectomemapper3/issues. (See *Contribute to Connectome Mapper* for more details)

CHAPTER

SIX

FUNDING

Work supported by the SNF Sinergia Grant 170873 (http://p3.snf.ch/Project-170873).

BIBLIOGRAPHY

[Smith2013SIFT] R.E. Smith et al., NeuroImage 67 (2013), pp. 298–312, <https://www.ncbi.nlm.nih.gov/pubmed/23238430>.

[Smith2015SIFT2] Smith RE et al., Neuroimage, 2015, 119:338-51. <https://doi.org/10.1016/j.neuroimage.2015.06.092>.

[Iglesias2015Brainstem] Iglesias et al., NeuroImage, 113, June 2015, 184-195. <http://www.nmr.mgh.harvard.edu/~iglesias/pdf/Neuroimage_2015_brainstem.pdf>

[Iglesias2015Hippo] Iglesias et al., Neuroimage, 115, July 2015, 117-137. <http://www.nmr.mgh.harvard.edu/~iglesias/pdf/subfieldsNeuroimage2015preprint.pdf>

[Najdenovska18] Najdenovska et al., Sci Data 5, 180270 (2018). <https://doi.org/10.1038/sdata.2018.270>

PYTHON MODULE INDEX

INDEX

A

D

E

F

G

H

I

N

O

P

R

S

T

U

V

W