

---

# **CMP3 Documentation**

***Release v3.0.0-RC2***

**Brain Communication Pathways Sinergia Consortium**

**Jan 15, 2021**



# GETTING STARTED

<b>1</b>	<b>About</b>	<b>3</b>
<b>2</b>	<b>License information</b>	<b>5</b>
<b>3</b>	<b>Aknowledgment</b>	<b>7</b>
3.1	Help/Questions . . . . .	7
<b>4</b>	<b>Eager to contribute?</b>	<b>9</b>
<b>5</b>	<b>Contents</b>	<b>11</b>
5.1	Installation Instructions for Users . . . . .	11
5.2	Connectome Mapper 3 and the BIDS standard . . . . .	14
5.3	Commandline Usage . . . . .	15
5.4	Graphical User Interface . . . . .	18
5.5	Outputs of Connectome Mapper 3 . . . . .	73
5.6	Packages and modules . . . . .	79
5.7	Adopting Datalad for collaboration . . . . .	223
5.8	Running on a cluster (HPC) . . . . .	227
5.9	BSD 3-Clause License . . . . .	229
5.10	Changes . . . . .	230
5.11	Citing . . . . .	235
5.12	Contributing to Connectome Mapper 3 . . . . .	236
5.13	Support, Bugs and New Feature Requests . . . . .	240
<b>6</b>	<b>Funding</b>	<b>241</b>
	<b>Bibliography</b>	<b>243</b>
	<b>Python Module Index</b>	<b>245</b>
	<b>Index</b>	<b>247</b>



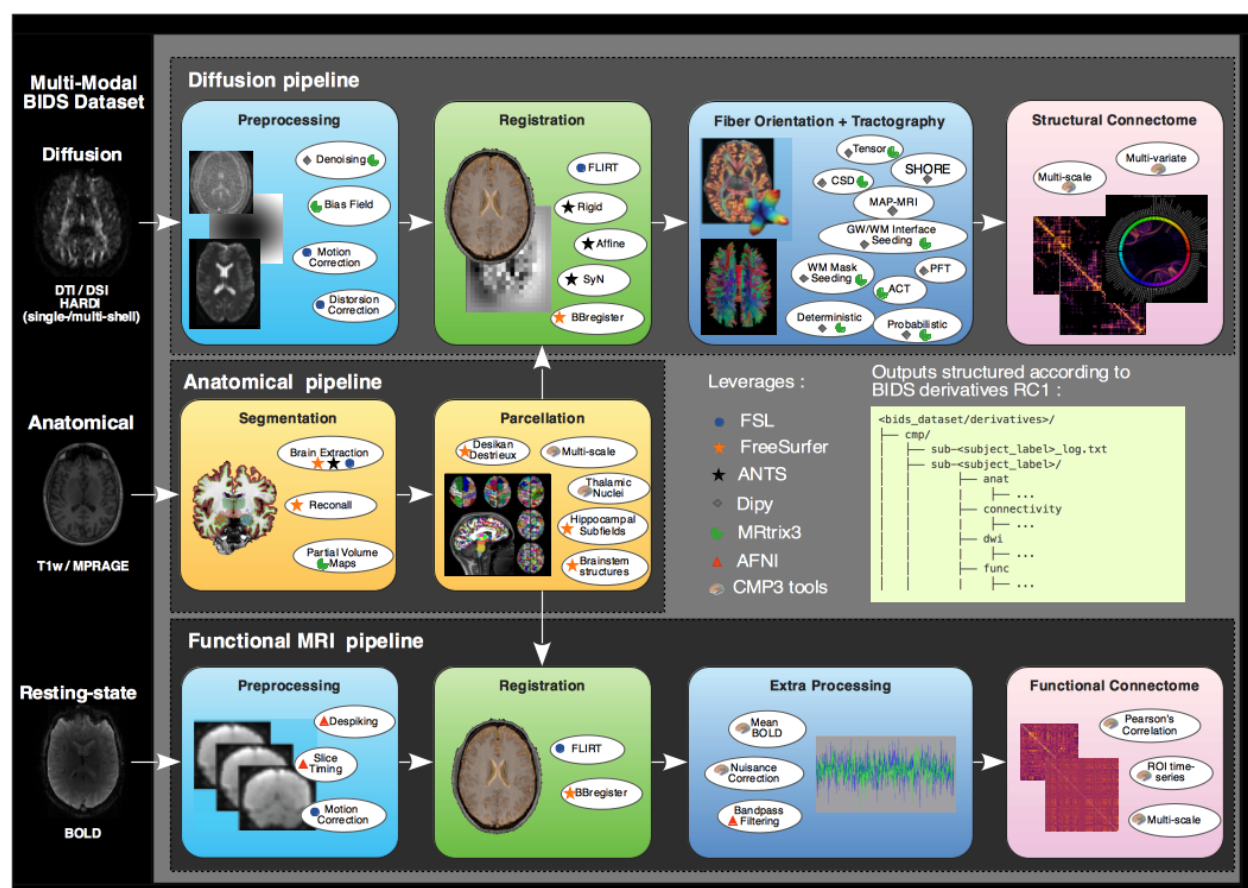


**Latest released version:** v3.0.0-RC2

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.





Connectome Mapper 3 implements full anatomical, diffusion and resting-state MRI processing pipelines, from raw Diffusion / T1 / T2 / BOLD data to multi-resolution connection matrices.

Connectome Mapper 3 pipelines use a combination of tools from well-known software packages, including FSL, FreeSurfer, ANTs, MRtrix3, Dipy and AFNI, orchestrated by the Nipype dataflow library. These pipelines were designed to provide the best software implementation for each state of processing, and will be updated as newer and better neuroimaging software become available.

This tool allows you to easily do the following:

- Take T1 / Diffusion / resting-state MRI data from raw to multi-resolution connection matrices.
- Implement tools from different software packages.
- Achieve optimal data processing quality by using the best tools available

- Automate and parallelize processing steps, which provides a significant speed-up from typical linear, manual processing.

Reproducibility and replicability is achieved through the distribution of a BIDSApp, a software container image which provide a frozen environment where versions of all external softwares and libraries are fixed.

## 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.



## ACKNOWLEDGMENT

If you 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](#).





## 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.



## 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.

#### 5.1.1 The Connectome Mapper 3 BIDSApp

##### Prerequisites

- Installed Docker Engine corresponding to your system:
  - For Ubuntu 14.04/16.04/18.04, follow the instructions from the web page:

```
$ firefox https://docs.docker.com/install/linux/docker-ce/ubuntu/
```

- For Mac OSX (>=10.10.3), get the .dmg installer from the web page:

```
$ firefox https://store.docker.com/editions/community/docker-ce-desktop-mac
```

- For Windows (>=10), get the installer from the web page:

```
$ firefox https://store.docker.com/editions/community/docker-ce-desktop-  
↪ windows
```

**Note:** Connectome Mapper 3 BIDSApp has been tested only on Ubuntu and MacOSX. For Windows users, it might be required to make few patches in the Dockerfile.

- Docker managed 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
- Get the latest release (v3.0.0-RC2) of the BIDS App:  

```
$ docker pull sebastientourbier/connectomemapper-bidsapp:v3.0.0-RC2
```
- To display all docker images available:

```
$ docker images
```

You should see the docker image “connectomemapper-bidsapp” with tag “v3.0.0-RC2” 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

- Installed miniconda3 (Python 3) from the web page:

```
$ firefox https://conda.io/miniconda.html
```

Download the Python 3 installer corresponding to your 32/64bits MacOSX/Linux/Win system.

#### Installation

The installation of the Connectome Mapper 3 BIDS App Manager (CMPBIDSAPPManager) consists of a clone of the source code repository, the creation of conda environment with all python dependencies installed, and eventually the installation of the CMPBIDSAPPManager itself, as follows:

- Open a terminal
- Go to the folder in which you would like to clone the source code repository:

```
$ cd <INSTALLATION DIRECTORY>
```

- Clone the source code repository:

```
$ git clone https://github.com/connectomicslab/connectomemapper3.git
→ connectomemapper3
```

- Create a branch and checkout the code corresponding to this version release:

```
$ git fetch
$ git checkout tags/v3.0.0-RC2 -b v3.0.0-RC2
```

- Create a miniconda3 environment where all python dependencies will be installed, this by using the spec list “conda\_packages\_list.txt” provided by the repository:

```
$ conda env create -f connectomemapper3/environment.yml
```

---

**Important:** It seems there is no conda package for git-annex available on Mac. Git-annex should be installed on MacOSX using brew ([https://brew.sh/index\\_fr](https://brew.sh/index_fr)) i.e. `brew install git-annex`.

Note that git-annex is only necessary if you wish to use BIDS datasets managed by Datalad (<https://www.datalad.org/>), a very experimental feature. For the moment, I would not recommend to use right now as it has been a long time it has not been tested.

So, you can without any problem delete or comment the line `- git-annex=7.20190219` and all other lines related to datalad packages in the `environment.yml` and it should then work!

---

- Activate the conda environment:

```
$ source activate py37cmp-gui
```

or:

```
$ conda activate py37cmp-gui
```

- Install the Connectome Mapper BIDS App Manager from the Bash Shell using following commands:

```
(py37cmp-gui)$ cd connectomemapper3/
(py37cmp-gui)$ python setup_gui.py install
```

- You are ready to use the Connectome Mapper 3 BIDS App Manager. See the [dedicated user guide](#).

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

---

**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 MPAGE structural image and a DWI and/or resting-state fMRI image. 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.

### 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 v3.0.0-RC2

```
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]
                        [--number_of_threads NUMBER_OF_THREADS]
                        [--number_of_participants_processed_in_parallel NUMBER_OF_
                        ↪PARTICIPANTS_PROCESSED_IN_PARALLEL]
                        [--mrtrix_random_seed MRTRIX_RANDOM_SEED]
                        [--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

<b>bids_dir</b>	The directory with the input dataset formatted according to the BIDS standard.
<b>output_dir</b>	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.
<b>analysis_level</b>	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 .txt file for processing stages of the anatomical MRI processing pipeline
- dwi\_pipeline\_config** Configuration .txt file for processing stages of the diffusion MRI processing pipeline
- func\_pipeline\_config** Configuration .txt file for processing stages of the fMRI processing pipeline
- number\_of\_threads** The number of OpenMP threads used for multi-threading by Freesurfer, FSL, MRtrix3, Dipy, AFNI (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

To run the docker image in participant level mode (for one participant):

```
$ docker run -t --rm -u $(id -u):$(id -g) \
  -v /home/localadmin/data/ds001:/bids_dir \
  -v /media/localadmin/data/ds001/derivatives:/output_dir \
  (-v /usr/local/freesurfer/license.txt:/bids_dir/code/license.txt \
  →)
  sebastientourbier/connectomemapper3:v3.0.0-RC2 \
  /bids_dir /output_dir participant --participant_label 01 \
  →(--session_label 01 \)
    --anat_pipeline_config /bids_dir/code/ref_anatomical_config.
  →ini \)
    (--dwi_pipeline_config /bids_dir/code/ref_diffusion_config.ini \)
    (--func_pipeline_config /bids_dir/code/ref_fMRI_config.ini \)
    (--number_of_participants_processed_in_parallel 1)
```

---

**Note:** The local directory of the input BIDS dataset (here: /home/localadmin/data/ds001) and the output directory (here: /media/localadmin/data/ds001/derivatives) used to process have to be mapped to the folders /bids\_dir and /output\_dir respectively using the -v docker 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 with the -v docker run option a freesurfer license.txt, which can be located anywhere on its 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 inspection are facilitated through the use of the Connectome Mapper GUI, i.e. cmpbidsappmanager (see [dedicated documentation page](#))

---

### 5.3.3 Debugging

Logs are outputted into <output\_dir>/cmp/sub-<participant\_label>/sub-<participant\_label>\_log-cmpbidsapp.txt.

### 5.3.4 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>.

### 5.3.5 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](https://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 py27cmp-gui
```

or:

```
$ conda activate py27cmp-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
```

### 5.4.3 Load a BIDS dataset

The Connectome Mapper 3 BIDS App Manager allows you to:

- load a BIDS dataset stored locally

You only have to select the root directory of your valid BIDS dataset (see note below)

- create a new datalad/BIDS dataset locally from an existing local or remote datalad/BIDS dataset (This is a feature under development)

Select the mode “Install a Datalad/BIDS dataset”.

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 MPAGE structural image. We highly recommend that you validate your dataset with the free, online [BIDS Validator](#).

---

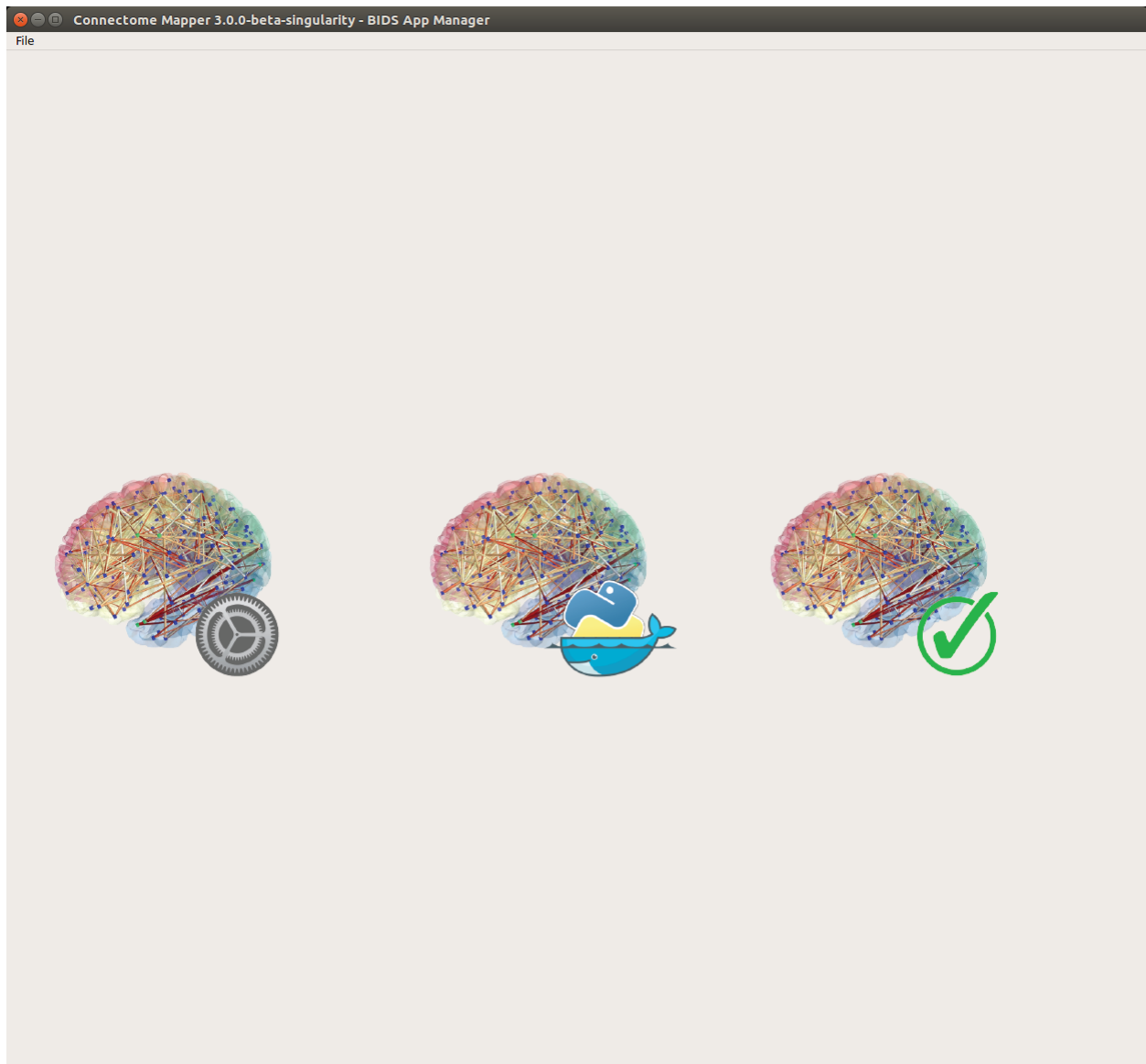
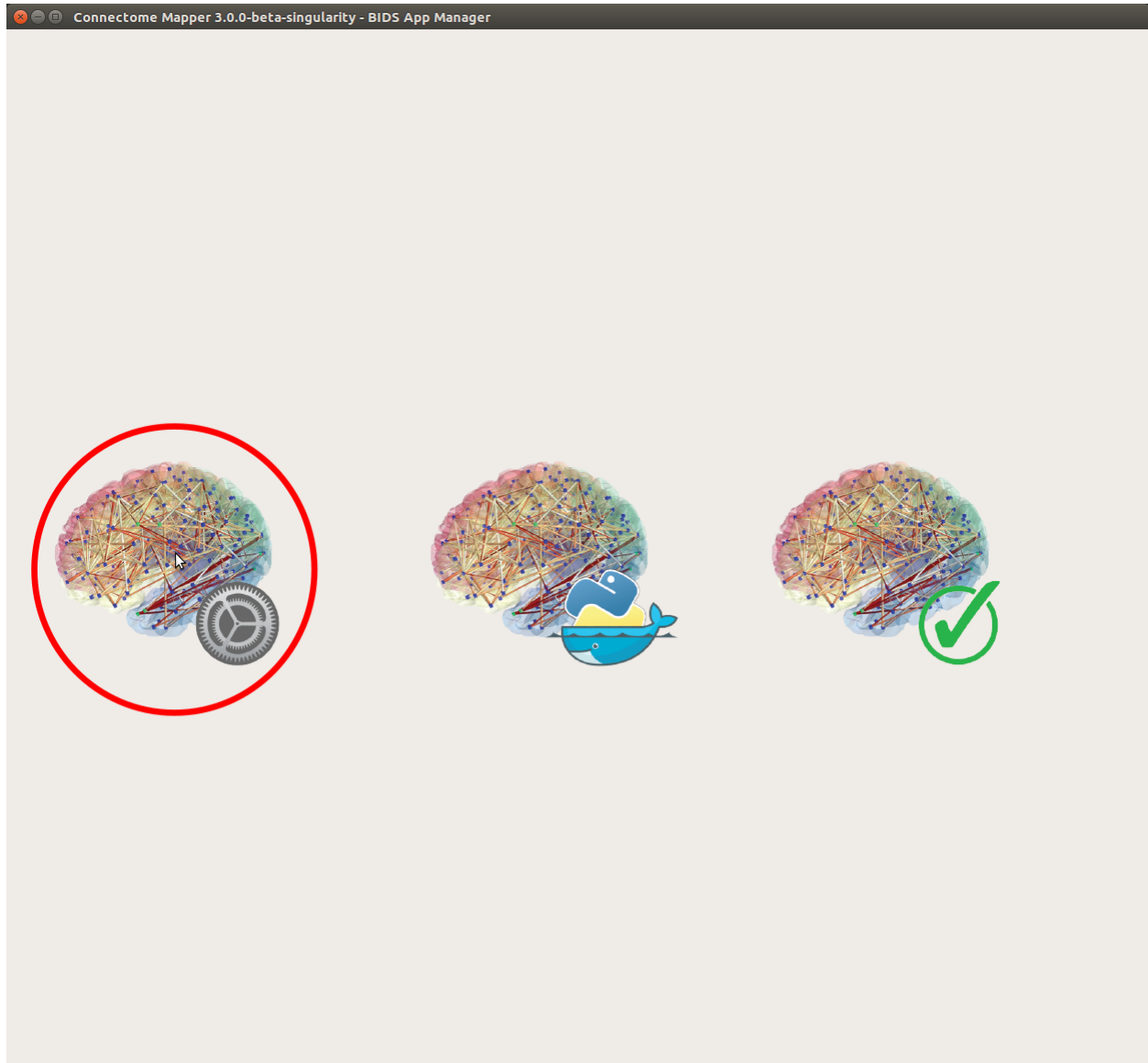


Fig. 1: Main window of the Connectome Mapper BIDS App Manager

## 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.

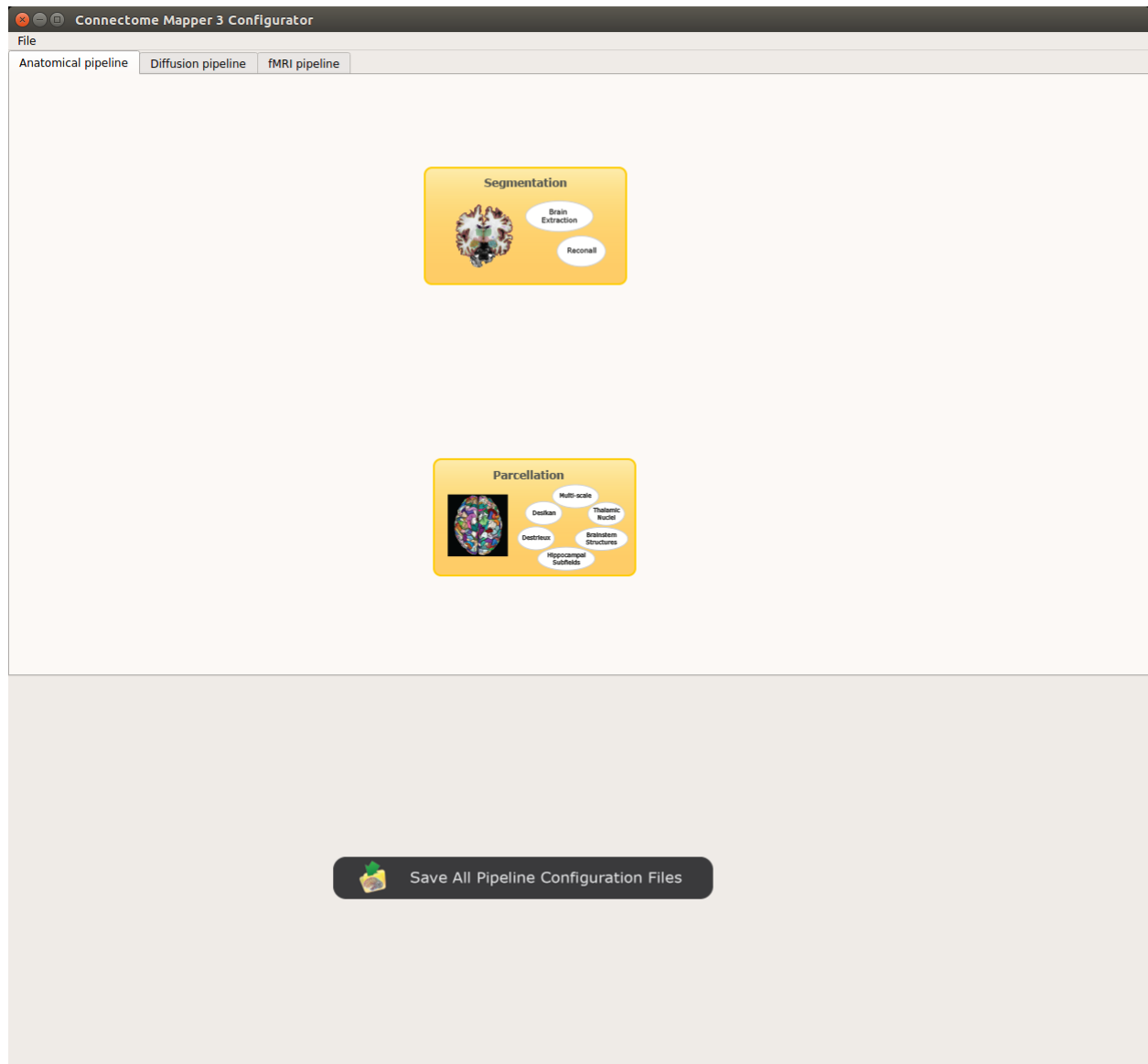


Fig. 2: Configurator Window of the Connectome Mapper

## Anatomical pipeline stages

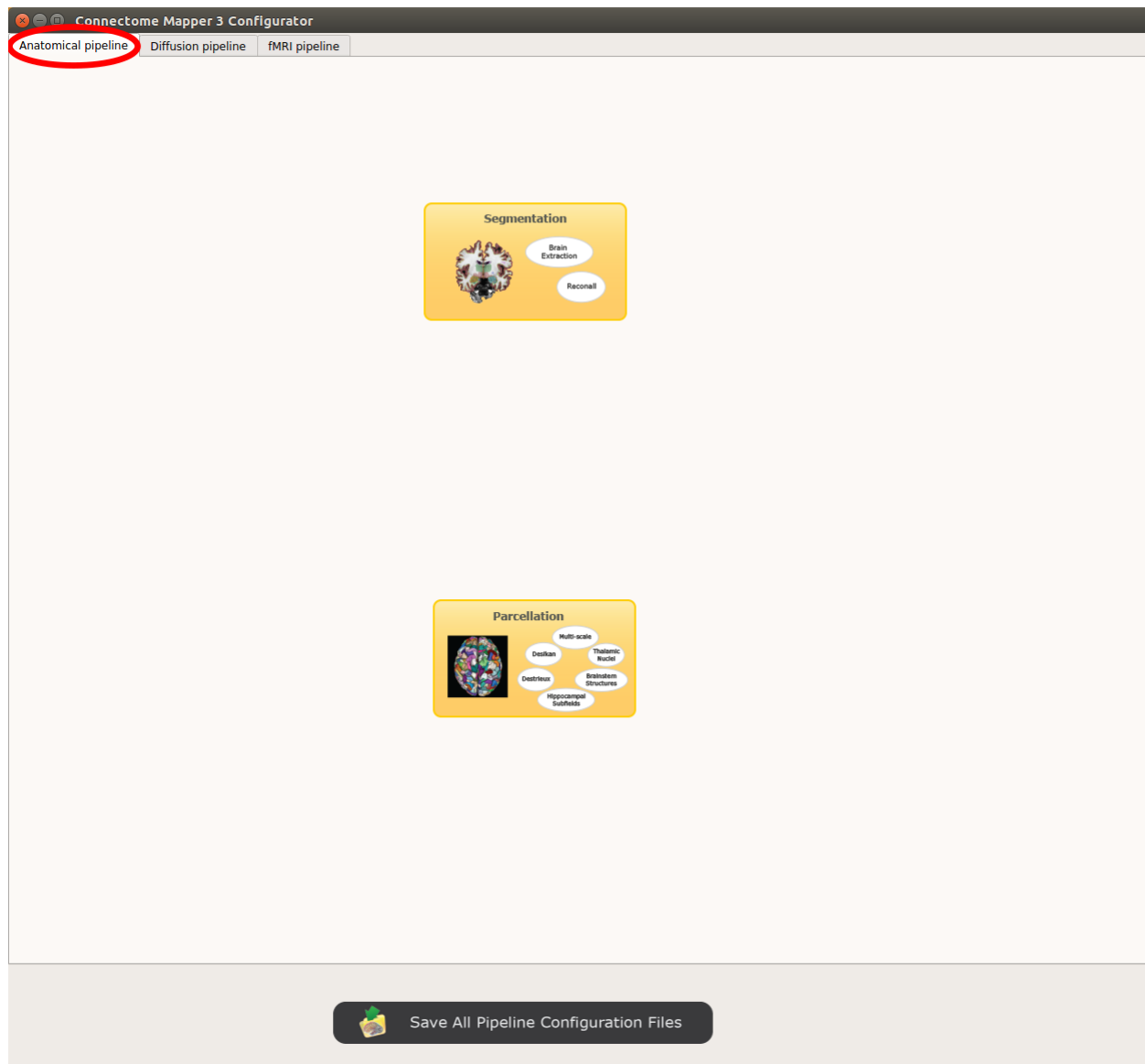
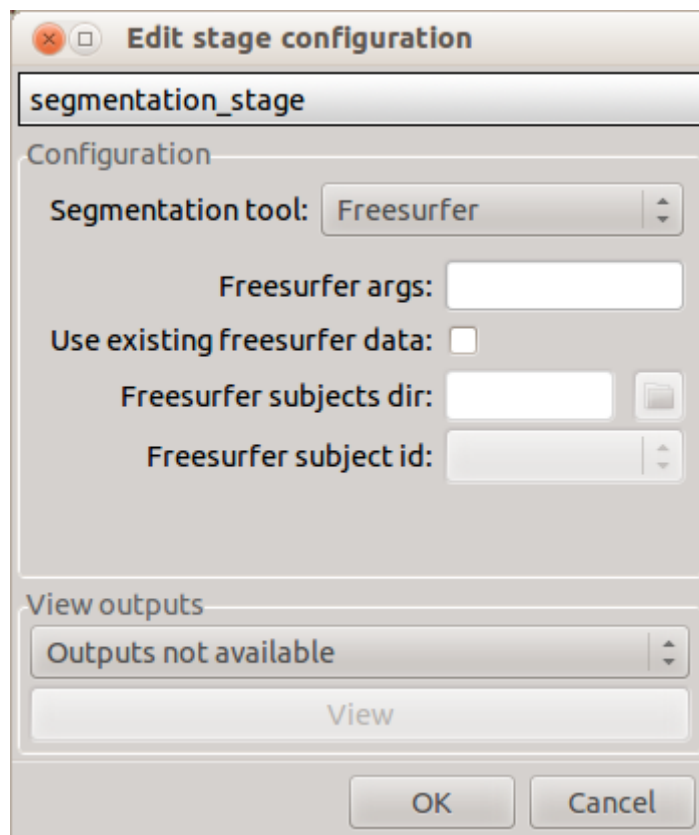


Fig. 3: Panel for configuration of anatomical pipeline stages

## Segmentation

Performs tissue segmentation using Freesurfer or custom segmentation.

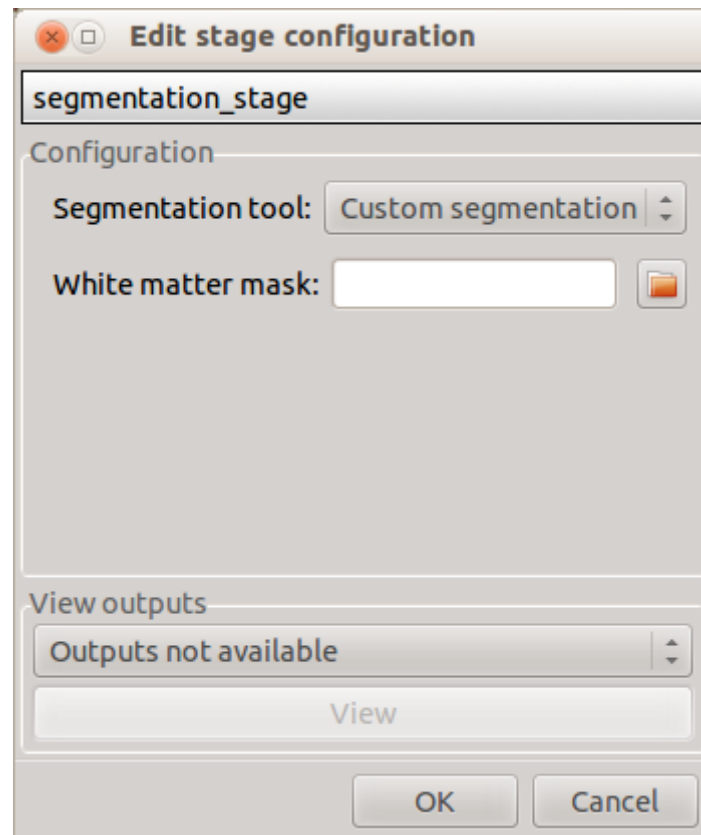
*Freesurfer*



- *Freesurfer args:* used to specify Freesurfer processing options
- *Use existing freesurfer data:* Check this box if you have already Freesurfer output data available

*Custom segmentation*

**Warning:** Not fully tested. Development and testing in progress.



- *White matter mask*: select the file containing your white matter binary mask

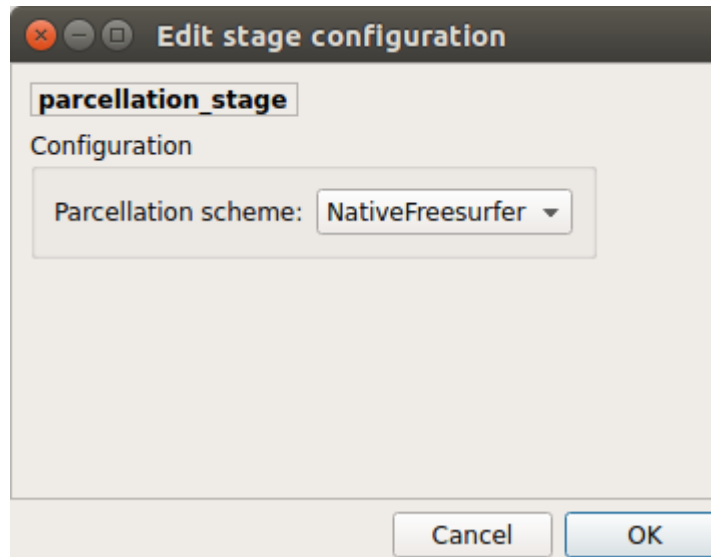
## Parcellation

Generates the Native Freesurfer or Lausanne2008/Lausanne2018 parcellation from Freesurfer data, or takes a custom parcellation atlas.

### Parcellation scheme

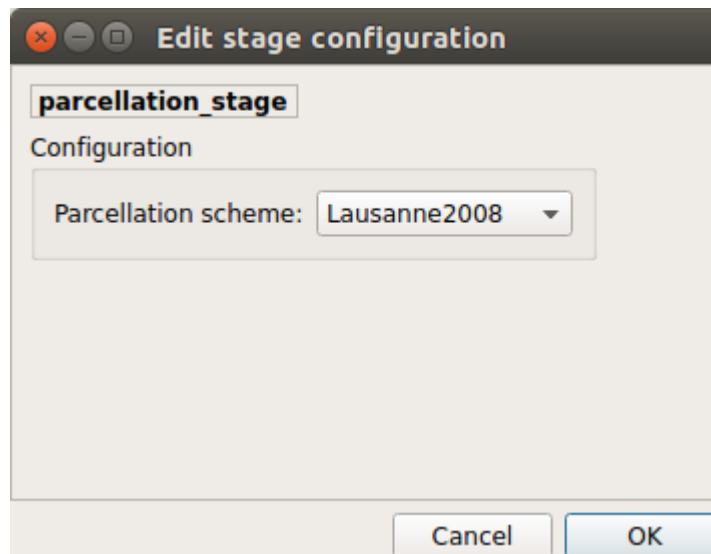
- *NativeFreesurfer*:





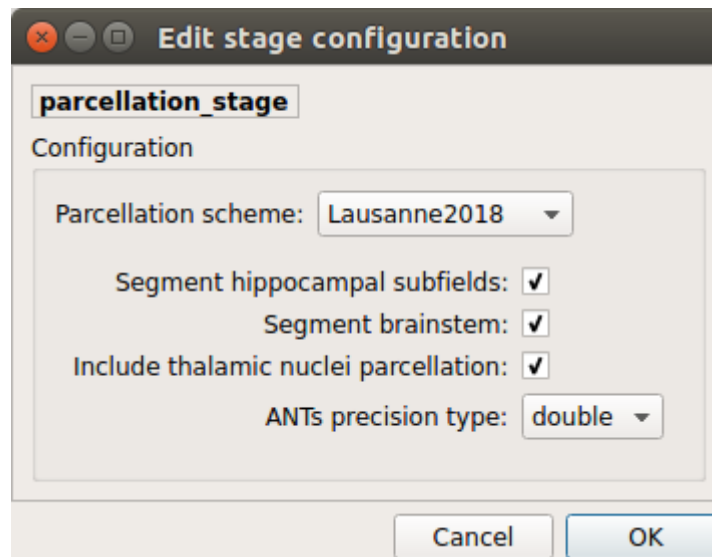
Atlas composed of 83 regions from the Freesurfer aparc+aseg file

- *Lausanne2008*:



Multi-resolution atlas

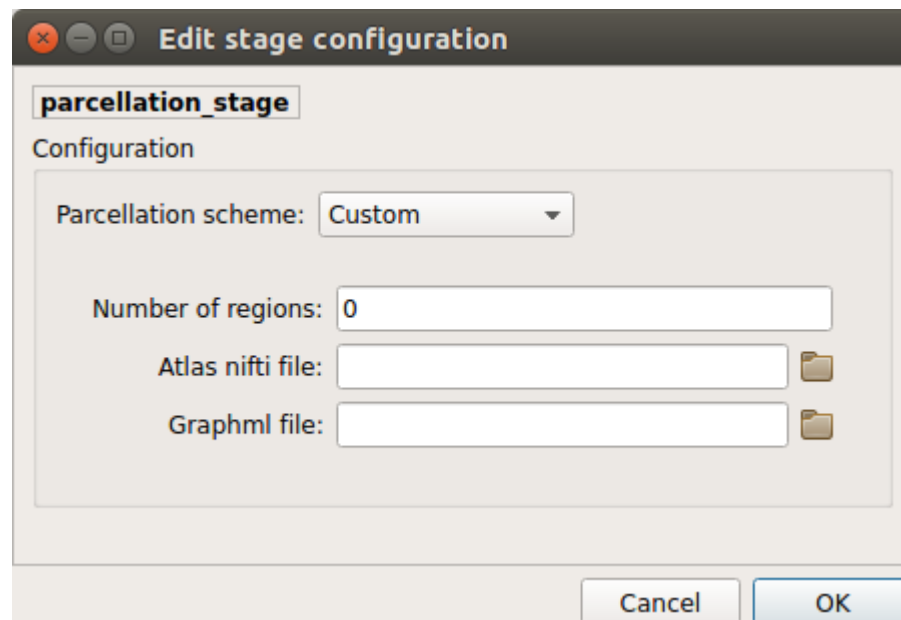
- *Lausanne2018*:



Lausanne 2008 atlas extended with 7 thalamic nuclei, 12 hippocampal subfields, and 4 brainstem sub-structure per hemisphere

- *Custom:*

**Warning:** Not fully tested. Development and testing in progress.



Custom atlas. Specify the atlas name, the number of regions, the nifti file and a corresponding graphml file. The Graphml file must contain at least a “dn\_correspondence\_id” field for each node. This field should contain the region’s label in the nifti file.

## Diffusion pipeline stages

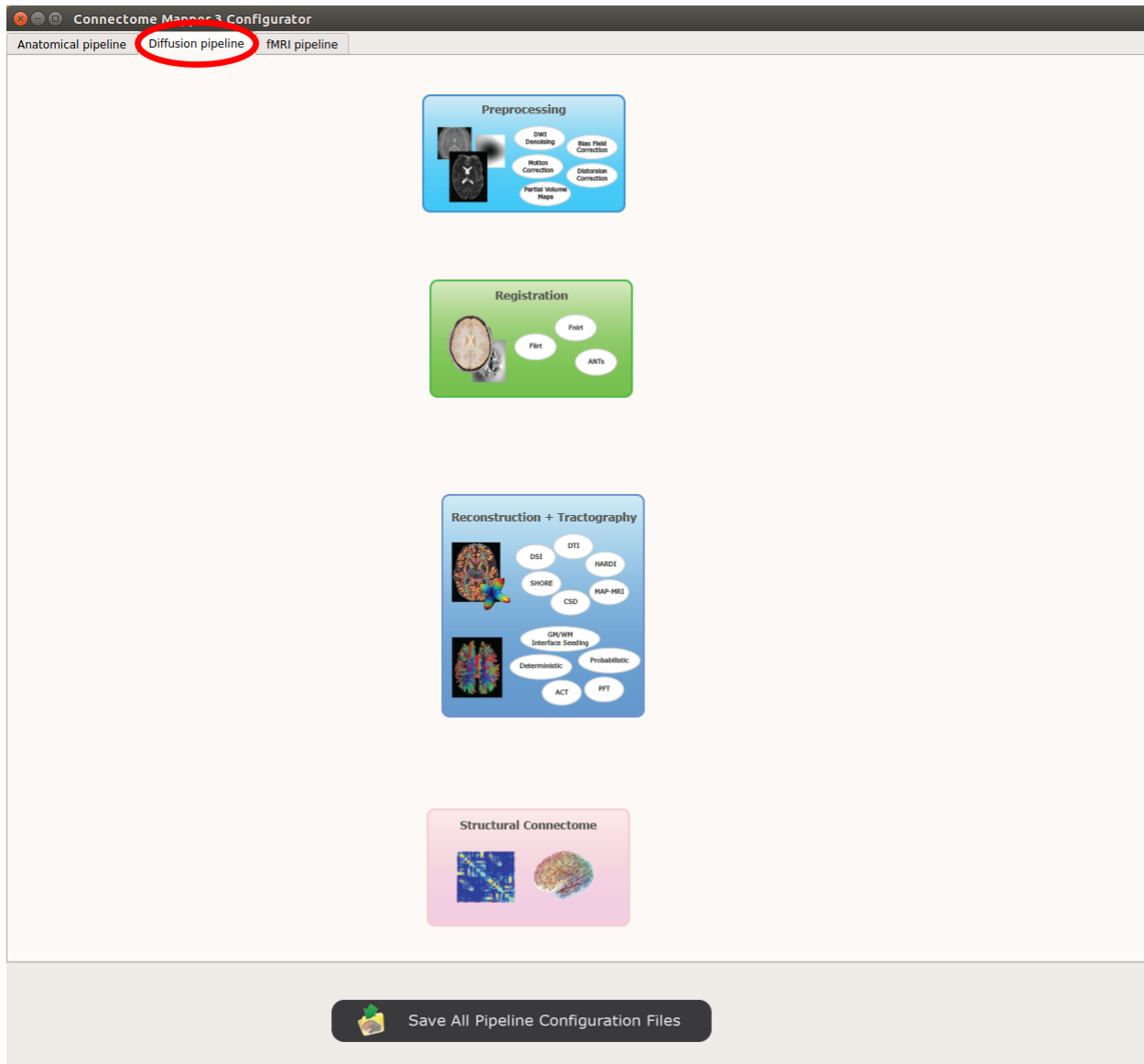


Fig. 4: Panel for configuration of diffusion pipeline stages

## Preprocessing

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

The screenshot shows a window titled "Edit stage configuration" with a tab labeled "preprocessing\_stage". The window contains a "Configuration" section with two main groups of settings:

- Preprocessing steps:**
  - Denoising: ☒ Tool: **Dipy (NLM)** Noise model (Dipy): **Rician**
  - Bias field correction: ☒ Tool: **ANTS N4**
  - Eddy current and motion correction: ☒ Eddy correction algo: **FSL eddy\_correct**
  - Motion correction: ☒
- Final resampling:**
  - Voxel size (x,y,z): F0: **1** F1: **1** F2: **1**
  - Interpolation: **interpolate**

At the bottom right of the window are "Cancel" and "OK" buttons.

### *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.

---

**Note:** For hemi-sphere DSI acquisitions, warning outputs will be displayed in the console when processing empty volumes.

---

### *Eddy current correction*

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

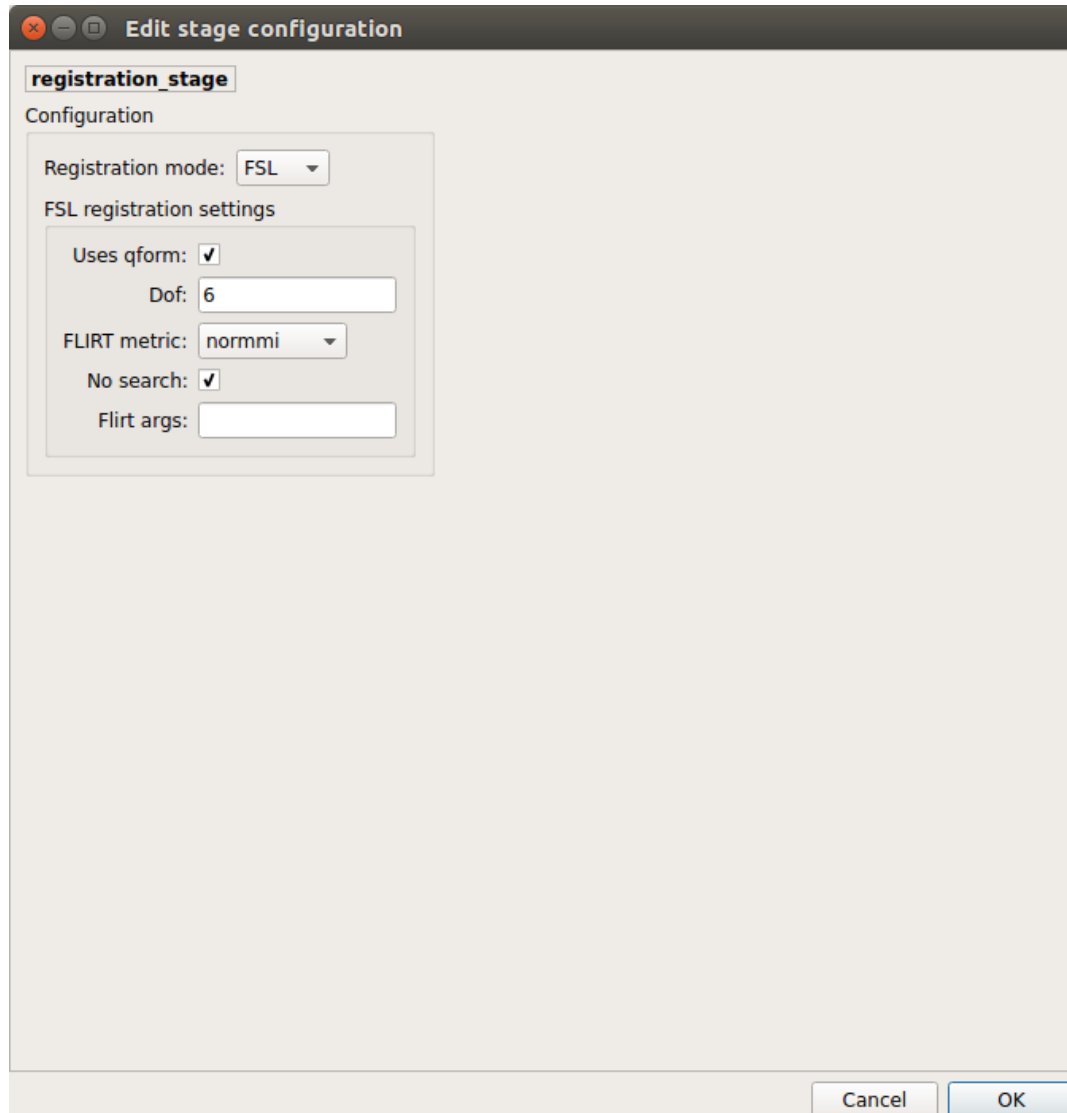
### *Resampling*

Resample morphological and diffusion data to F0 x F1 x F2 mm<sup>3</sup>

## Registration

### Registration mode

- FSL (Linear):



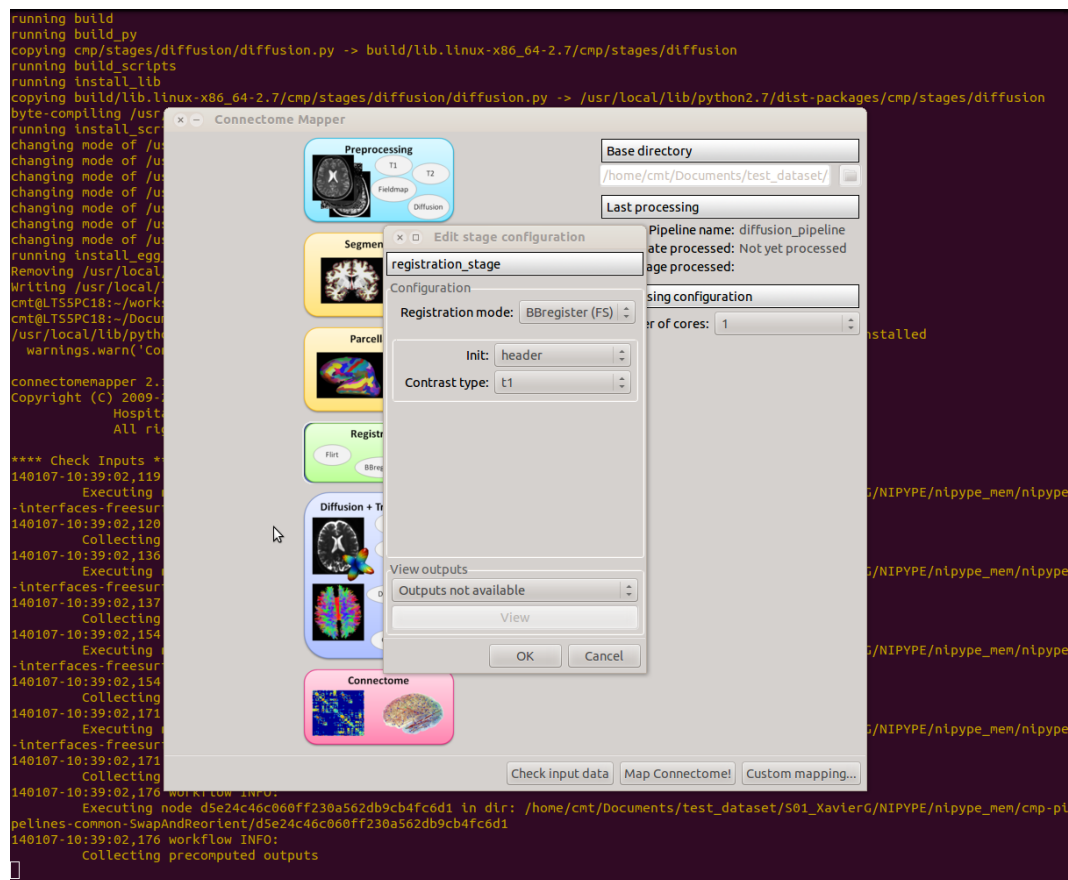
The screenshot shows a window titled "Edit stage configuration" with a tab labeled "registration\_stage". The window contains a "Configuration" section with the following settings:

- Registration mode: FSL (selected in a dropdown menu)
- FSL registration settings:
  - Uses qform: ☒
  - Dof: 6 (text input)
  - FLIRT metric: normmi (selected in a dropdown menu)
  - No search: ☒
  - Flirt args: (empty text input)

At the bottom right of the window are "Cancel" and "OK" buttons.

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

- BBregister (FS):



Perform linear registration using Freesurfer BBregister tool.

- Non-linear (ANTS):

**Edit stage configuration**

**registration\_stage**

Configuration

Registration mode: **ANTs**

ANTs registration settings

**General**

Interpolation: **BSpline** Parameters: **5**

winsorize lower quantile: **0.005** winsorize upper quantile: **0.995**

Convergence threshold: **1e-06** Convergence window size: **10**

Use float precision to save memory: ☐

**Rigid + Affine**

Metric: **MI**

Gradient step size: **0.1**

Sampling strategy: **Regular** Sampling percentage: **0.25**

Gradient step size: **0.1**

Symmetric diffeomorphic SyN registration: ☒

**SyN (symmetric diffeomorphic registration)**

Metric: **CC**

Gradient step size: **0.1**

Update field variance in voxel space: **3.0**

Total field variance in voxel space: **0.0**

**Cancel** **OK**

Perform symmetric diffeomorphic SyN registration from T1 to b0

## 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:

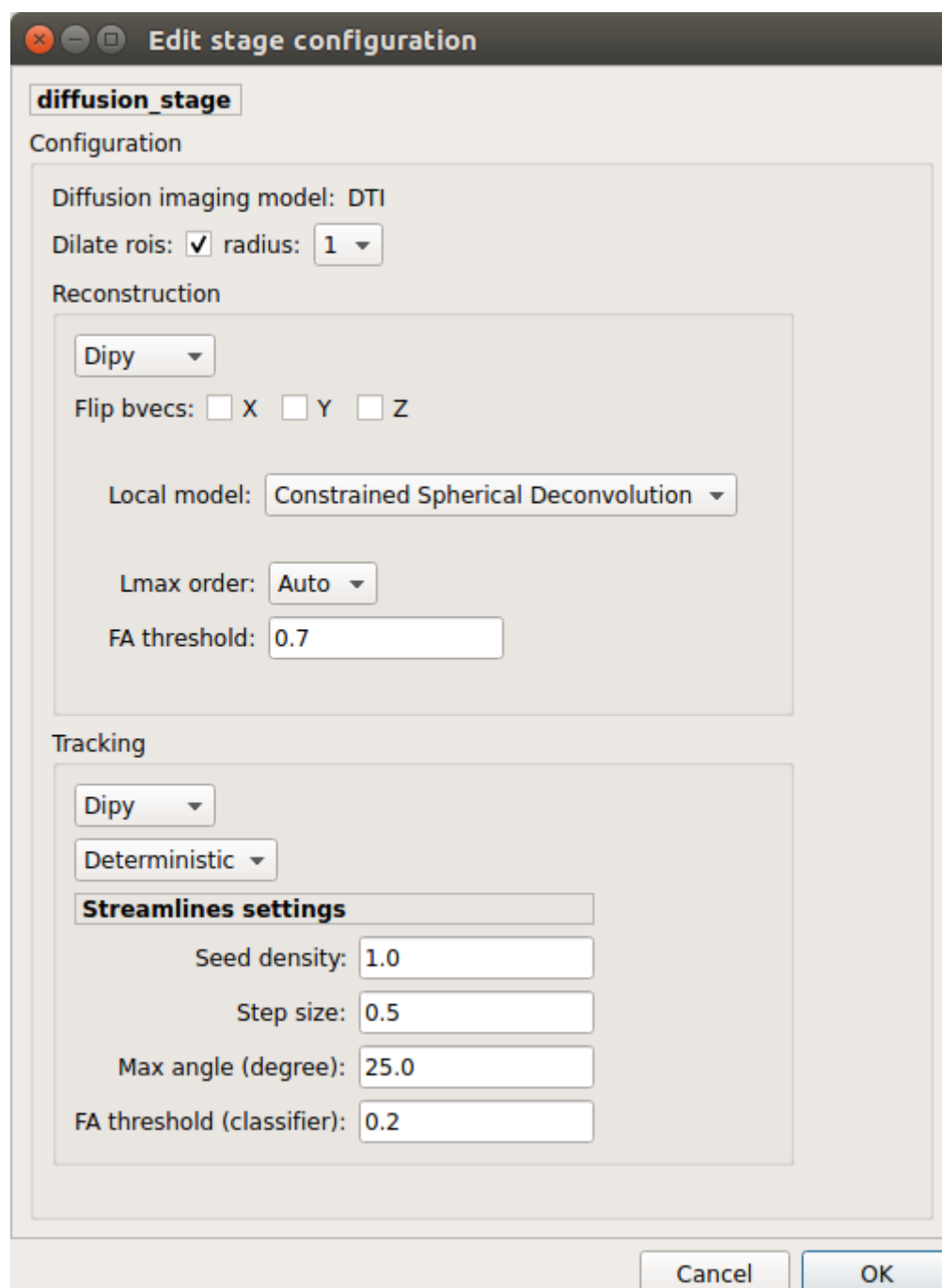


Fig. 5: Diffusion stage configuration window



Reconstruction

Dipy

Flip bvecs: ☐ X ☐ Y ☐ Z

**Parameters of SHORE reconstruction model**

Radial order: 6

Scale factor (zeta): 700

Radial regularization constant: 1e-08

Angular regularization constant: 1e-08

Diffusion time (s): 0.0253302959106

Constrain the optimization such that  $E(0) = 1$ . ☐

Constrain the propagator to be positive. ☐

Mapmri: ☐

SHORE performed only on DSI data

- Tensor:

Reconstruction

Dipy

Flip bvecs: ☐ X ☐ Y ☐ Z

Local model: Tensor

Tensor performed only on DTI data

- CSD:

Reconstruction

Dipy

Flip bvecs: ☐ X ☐ Y ☐ Z

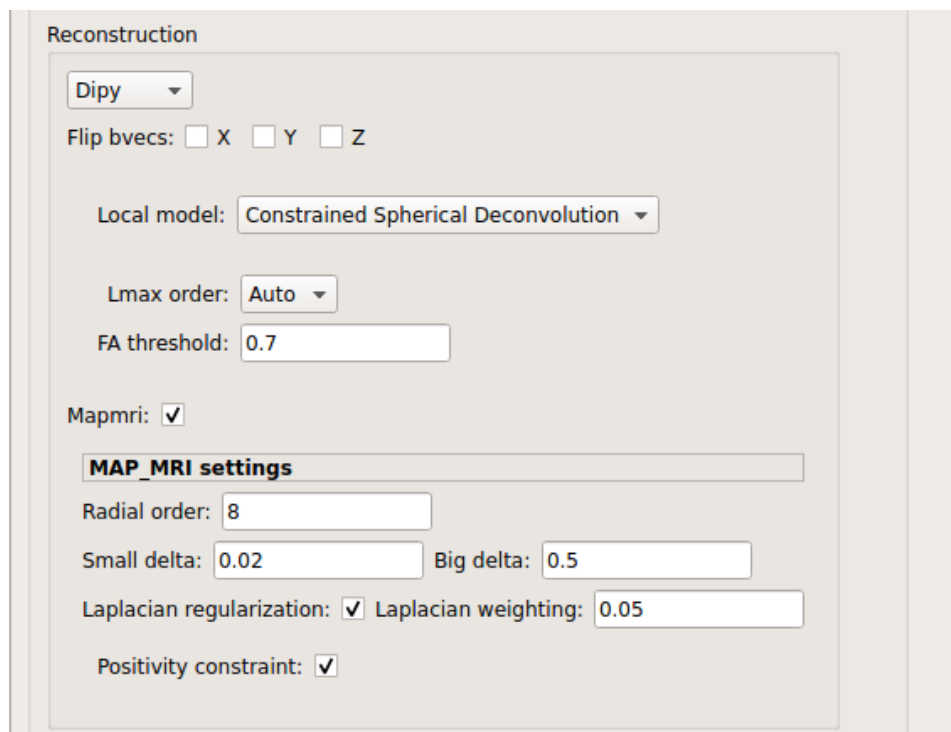
Local model: Constrained Spherical Deconvolution

Lmax order: Auto

FA threshold: 0.7

CSD performed on DTI and multi-shell data

- MAP\_MRI:

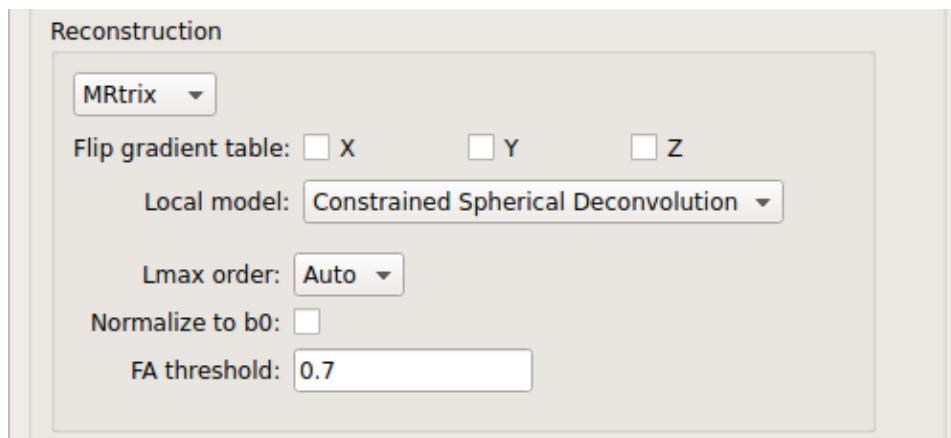


The screenshot shows the 'Reconstruction' panel in a software interface. At the top, a dropdown menu is set to 'Dipy'. Below it are three checkboxes for 'Flip bvecs' labeled X, Y, and Z, all of which are unchecked. The 'Local model' is set to 'Constrained Spherical Deconvolution' via a dropdown. 'Lmax order' is set to 'Auto' and 'FA threshold' is set to '0.7'. The 'Mapmri' checkbox is checked. A section titled 'MAP\_MRI settings' contains a 'Radial order' set to '8', 'Small delta' set to '0.02', 'Big delta' set to '0.5', 'Laplacian regularization' checked, 'Laplacian weighting' set to '0.05', and 'Positivity constraint' checked.

MAP-MRI performed only on multi-shell data

**MRtrix:** perform CSD reconstruction.

- CSD:



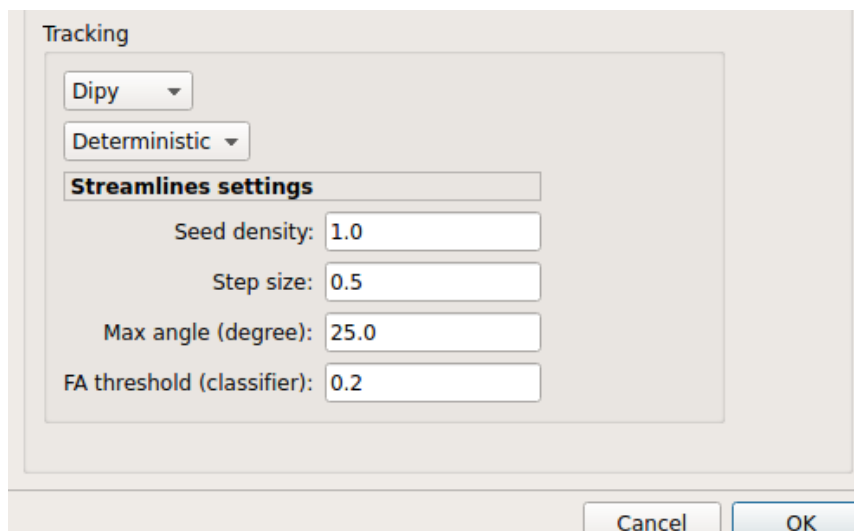
The screenshot shows the 'Reconstruction' panel with the dropdown menu set to 'MRtrix'. The 'Flip gradient table' checkboxes for X, Y, and Z are all unchecked. The 'Local model' remains 'Constrained Spherical Deconvolution'. 'Lmax order' is 'Auto'. The 'Normalize to b0' checkbox is unchecked. The 'FA threshold' is set to '0.7'.

CSD performed on DTI and multi-shell data

### Tractography tool

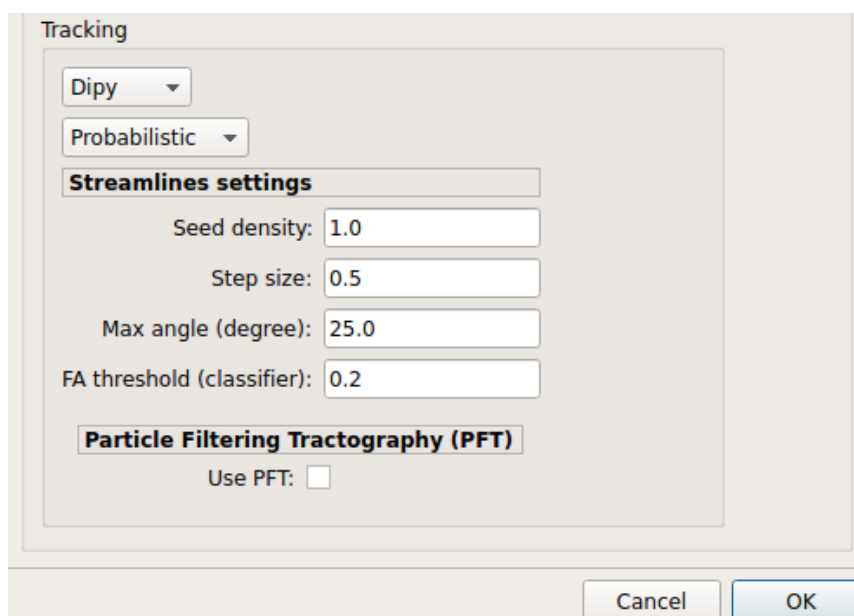
**Dipy:** perform deterministic and probabilistic fiber tracking as well as particle filtering tractography.

- Deterministic tractography:



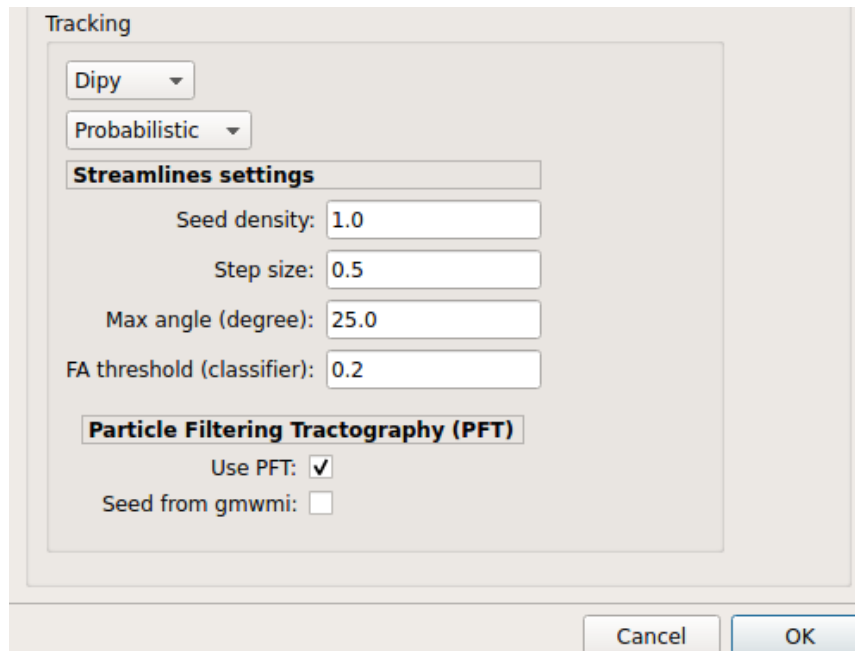
Deterministic tractography (SD\_STREAM) performed on single tensor or CSD reconstruction

- Probabilistic tractography:



Probabilistic tractography (iFOD2) performed on SHORE or CSD reconstruction

- Probabilistic particle filtering tractography (PFT):

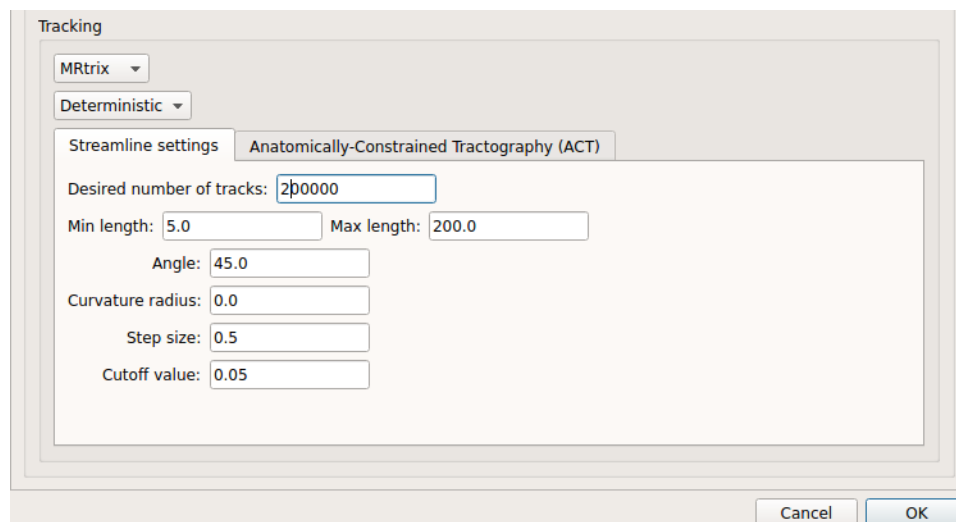


Probabilistic PFT tracking performed on SHORE or CSD reconstruction. Seeding from the gray matter / white matter interface is possible.

**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.

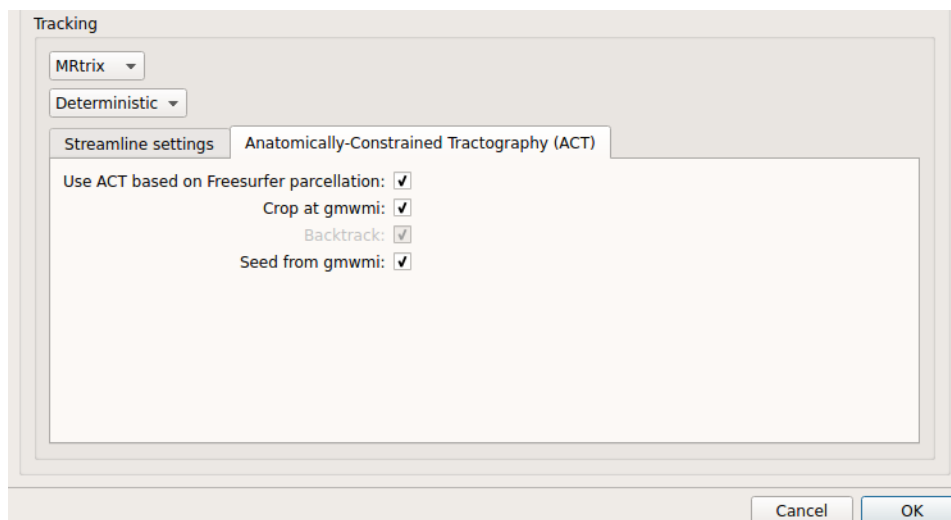
**MRtrix:** perform deterministic and probabilistic fiber tracking as well as anatomically-constrained tractography. ROI dilation is required to map brain connections when the tracking only operates in the white matter.

- Deterministic tractography:



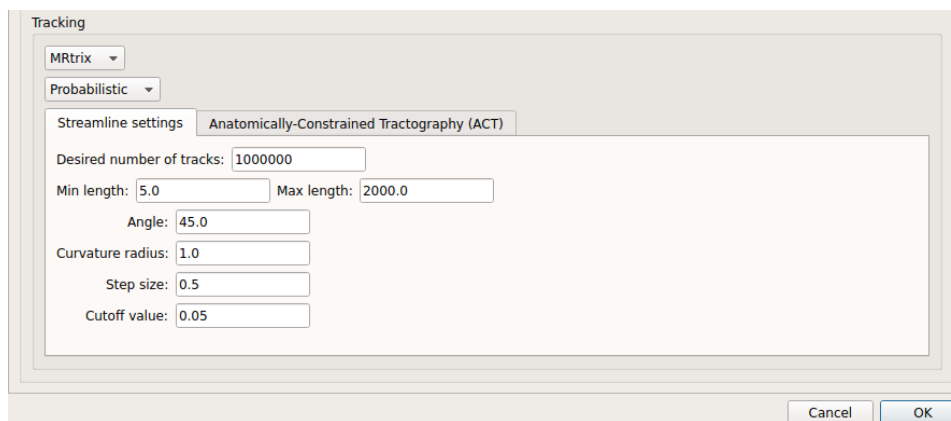
Deterministic tractography (SD\_STREAM) performed on single tensor or CSD reconstruction

- Deterministic anatomically-constrained tractography (ACT):



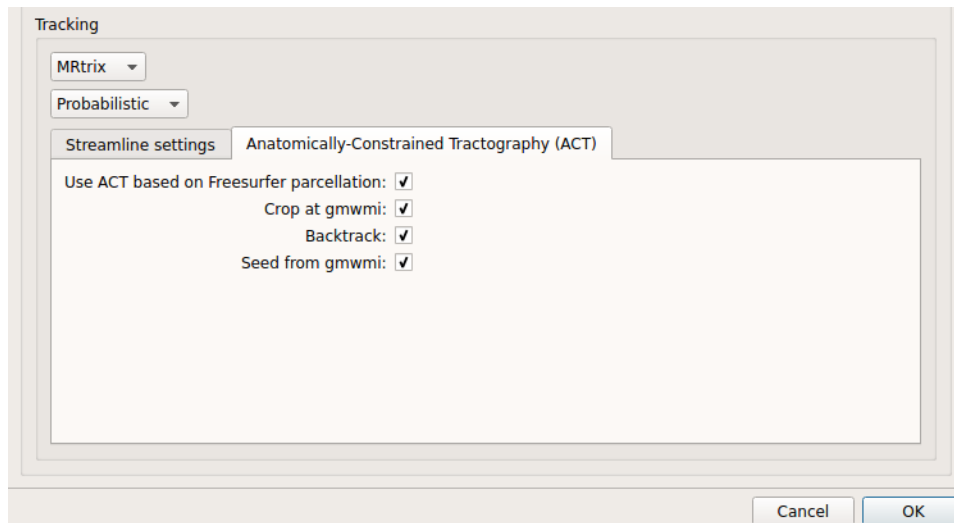
Deterministic ACT tracking performed on single tensor or CSD reconstruction. Seeding from the gray matter / white matter interface is possible. Backtrack option is not available in deterministic tracking.

- Probabilistic tractography:



Probabilistic tractography (iFOD2) performed on SHORE or CSD reconstruction

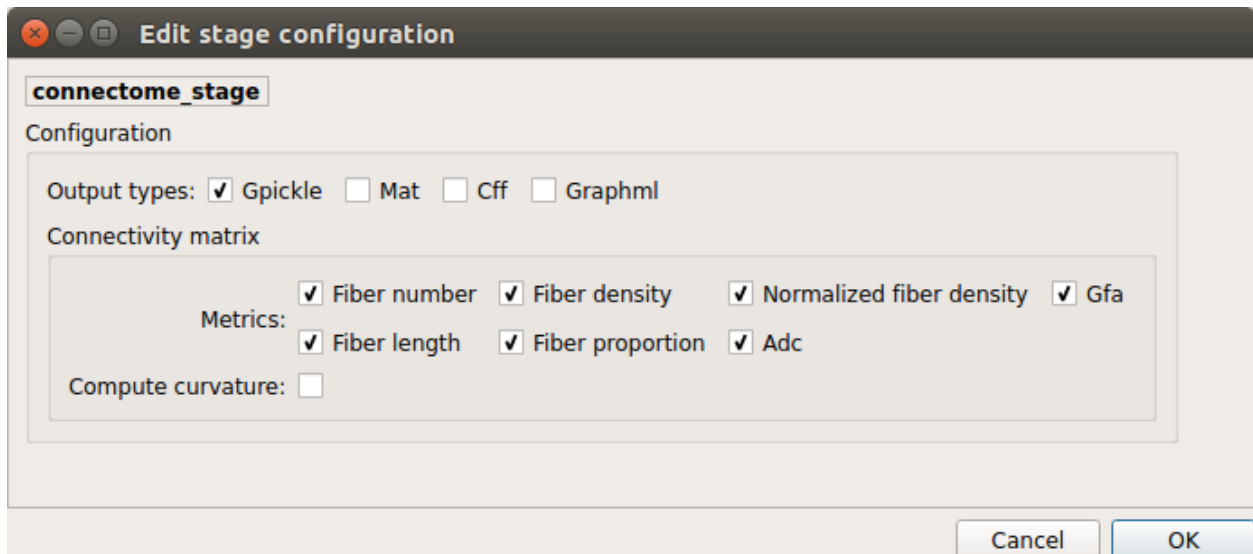
- Probabilistic anatomically-constrained tractography (ACT):



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.

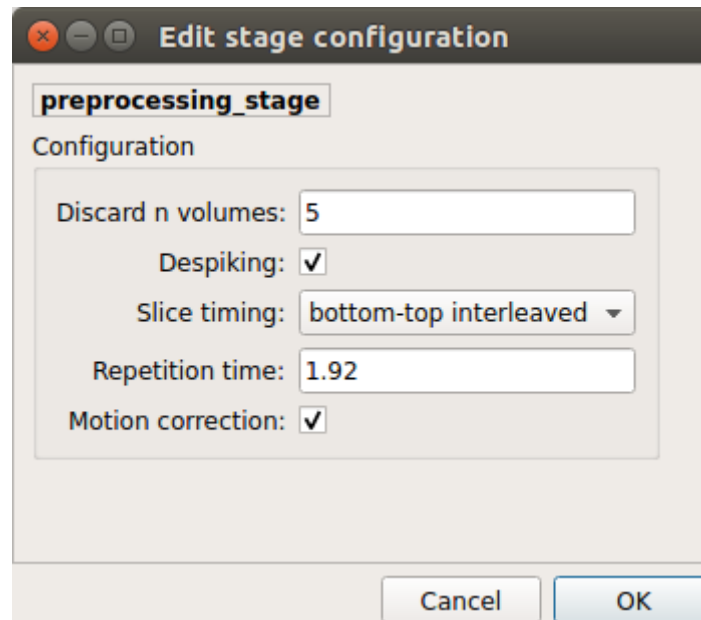
## FMRI pipeline stages



Fig. 6: Panel for configuration of 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 volumes*

Discard n volumes from further analysis

### *Despiking*

Perform despiking of the BOLD signal using AFNI.

### *Slice timing and Repetition time*

Perform slice timing correction using FSL's slicetimer.

### *Motion correction*

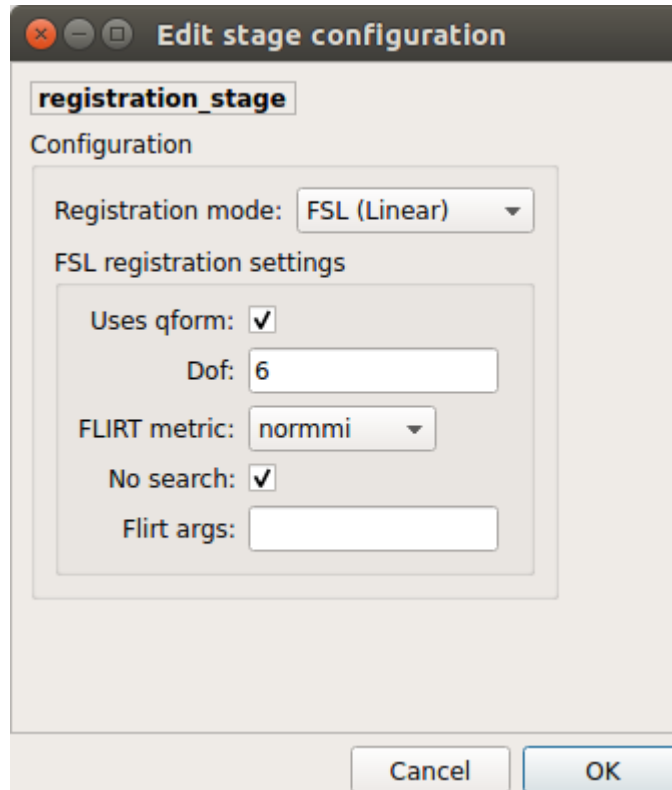
Align BOLD volumes to the mean BOLD volume using FSL's MCFLIRT.



## Registration

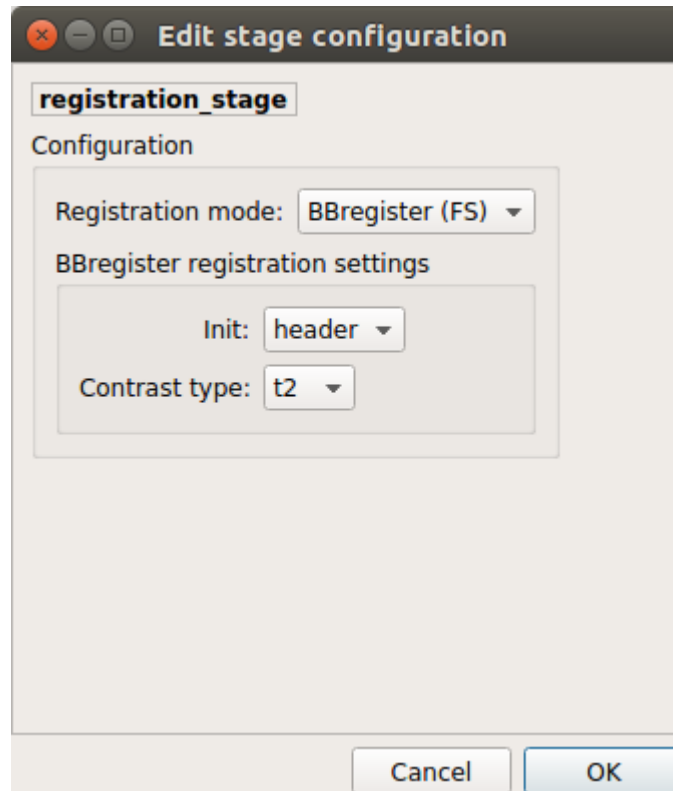
### Registration mode

- FSL (Linear):



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

- BBregister (FS)



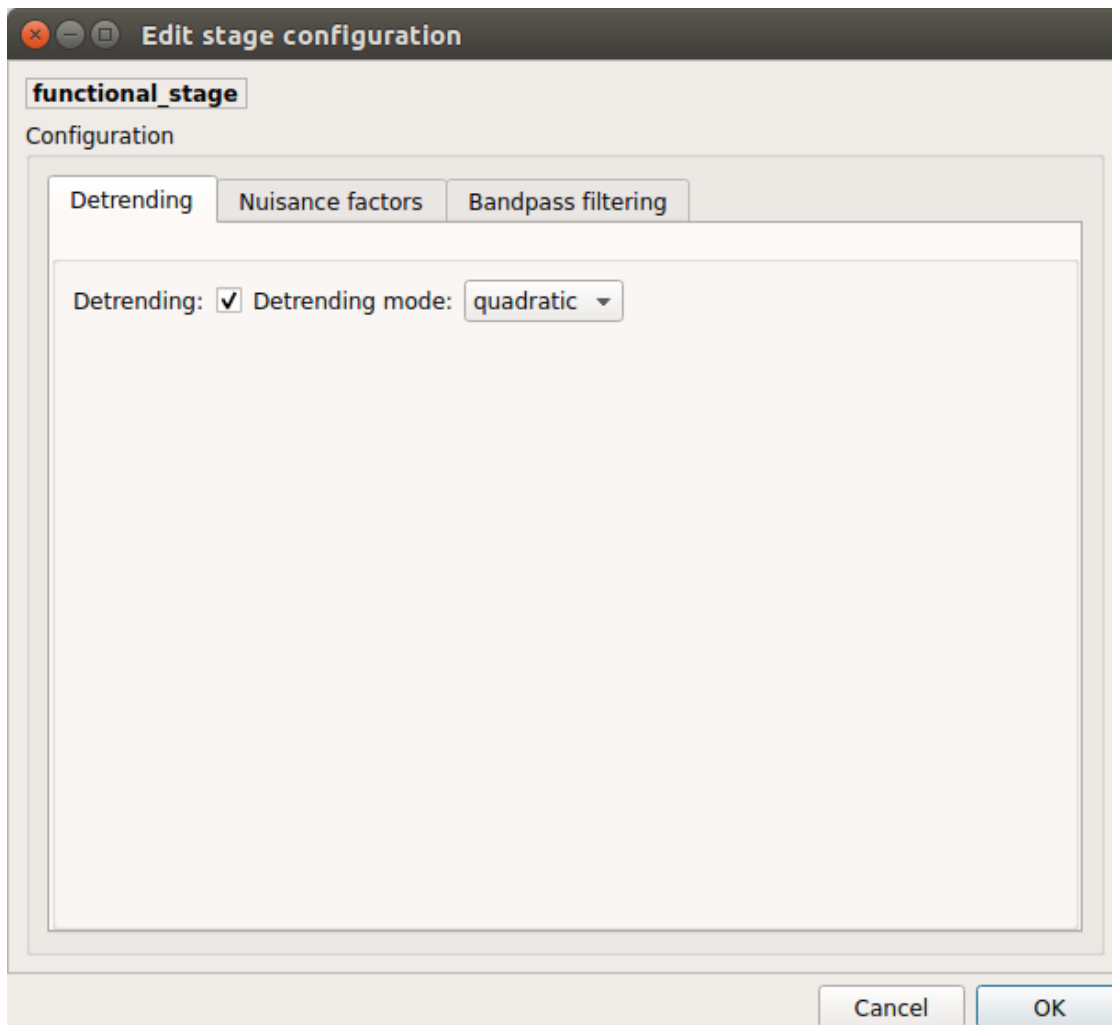
Perform linear registration using Freesurfer BBregister tool from T1 to mean BOLD via T2.

**Warning:** development in progress

## fMRI processing

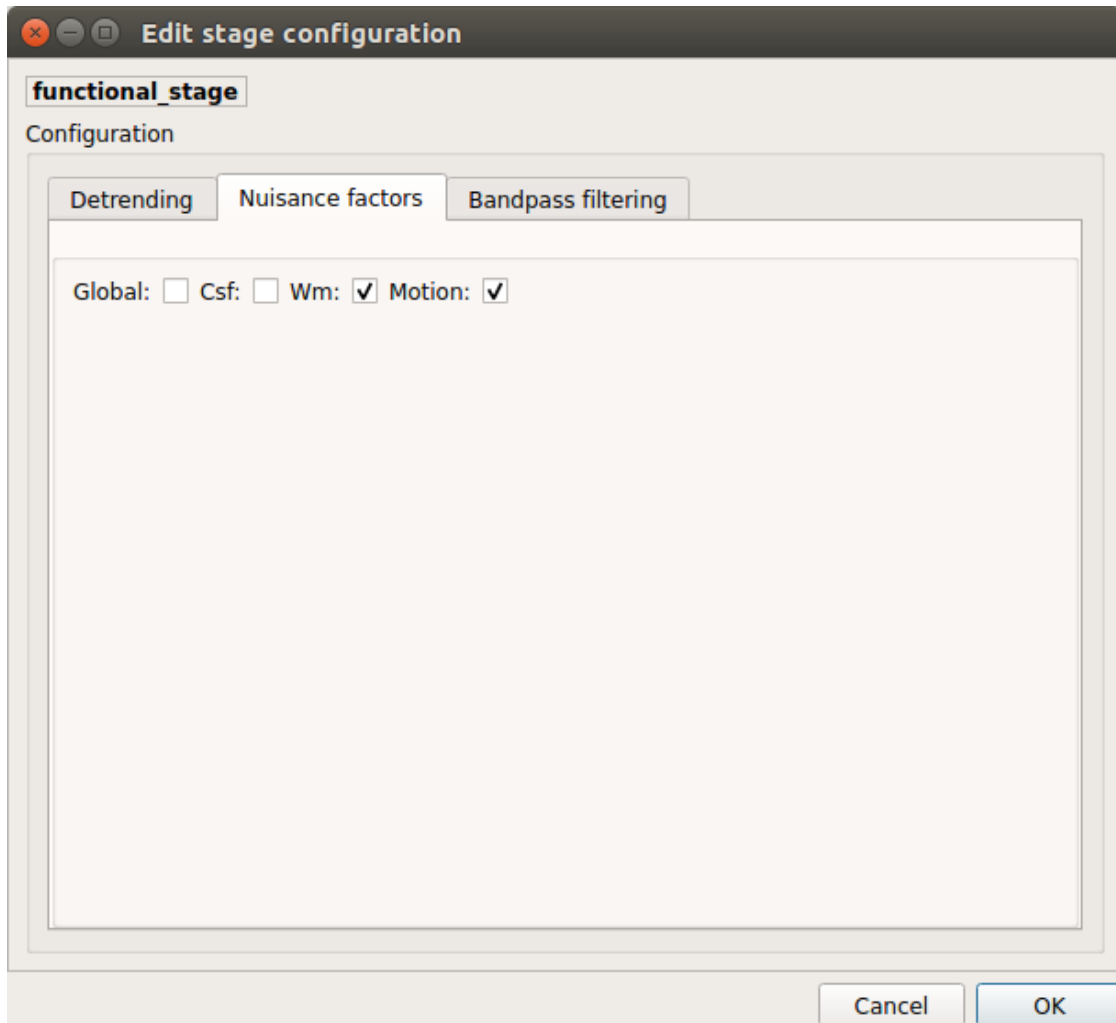
Performs detrending, nuisance regression, bandpass filtering, diffusion reconstruction and local deterministic or probabilistic tractography based on several tools. ROI dilation is required to map brain connections when the tracking only operates in the white matter.

*Detrending*



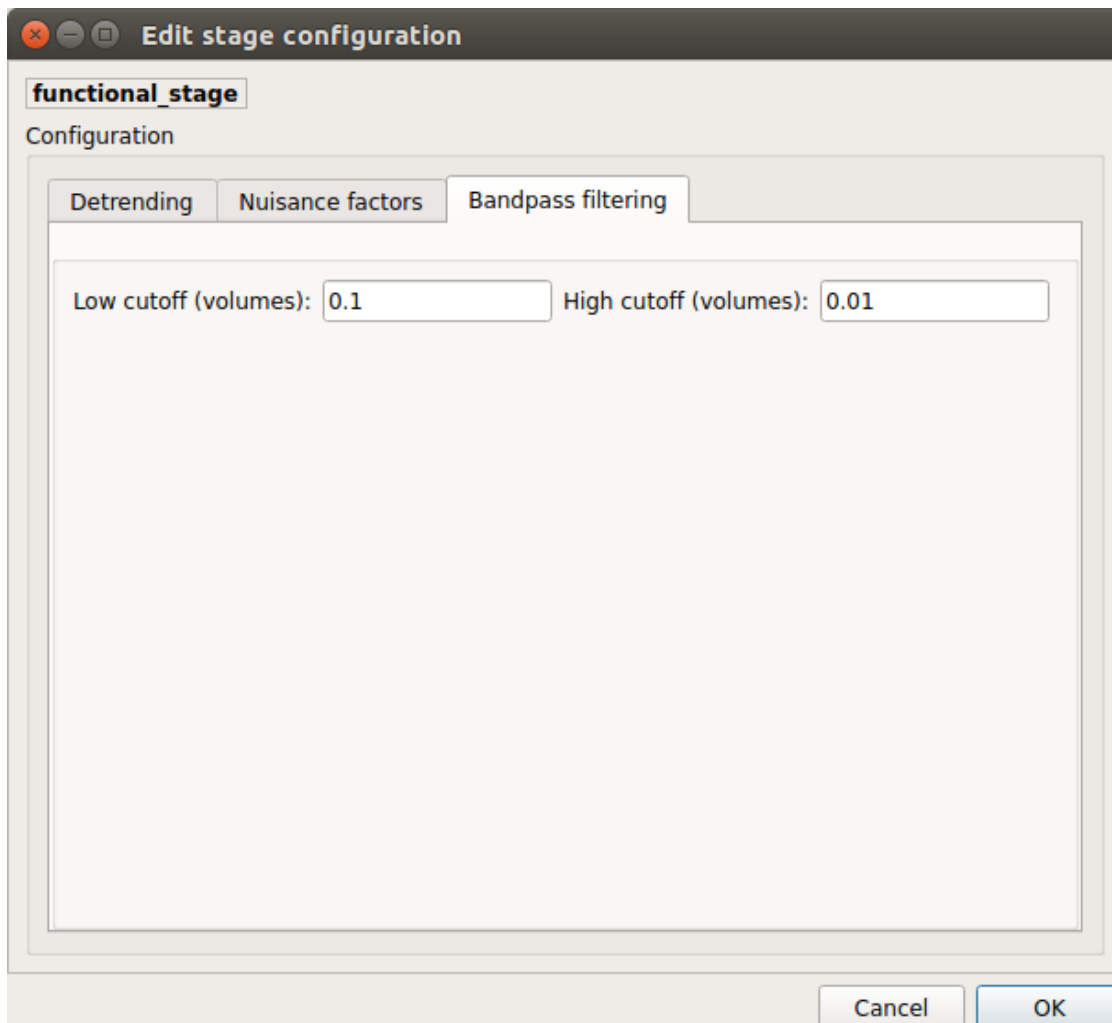
Detrending of BOLD signal using: 1. *linear* trend removal algorithm provided by the `scipy` library 2. *quadratic* trend removal algorithm provided by the `obspy` library

*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

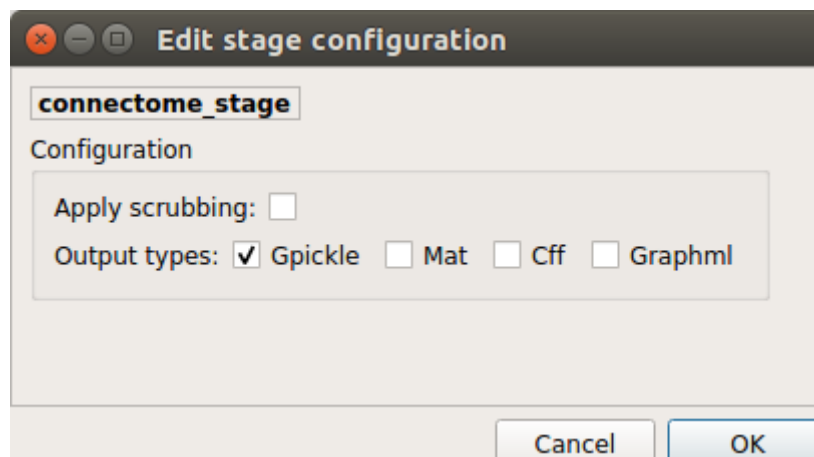
*Bandpass filtering*



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.

## **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` 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 configuration file as...)

## **Nipype**

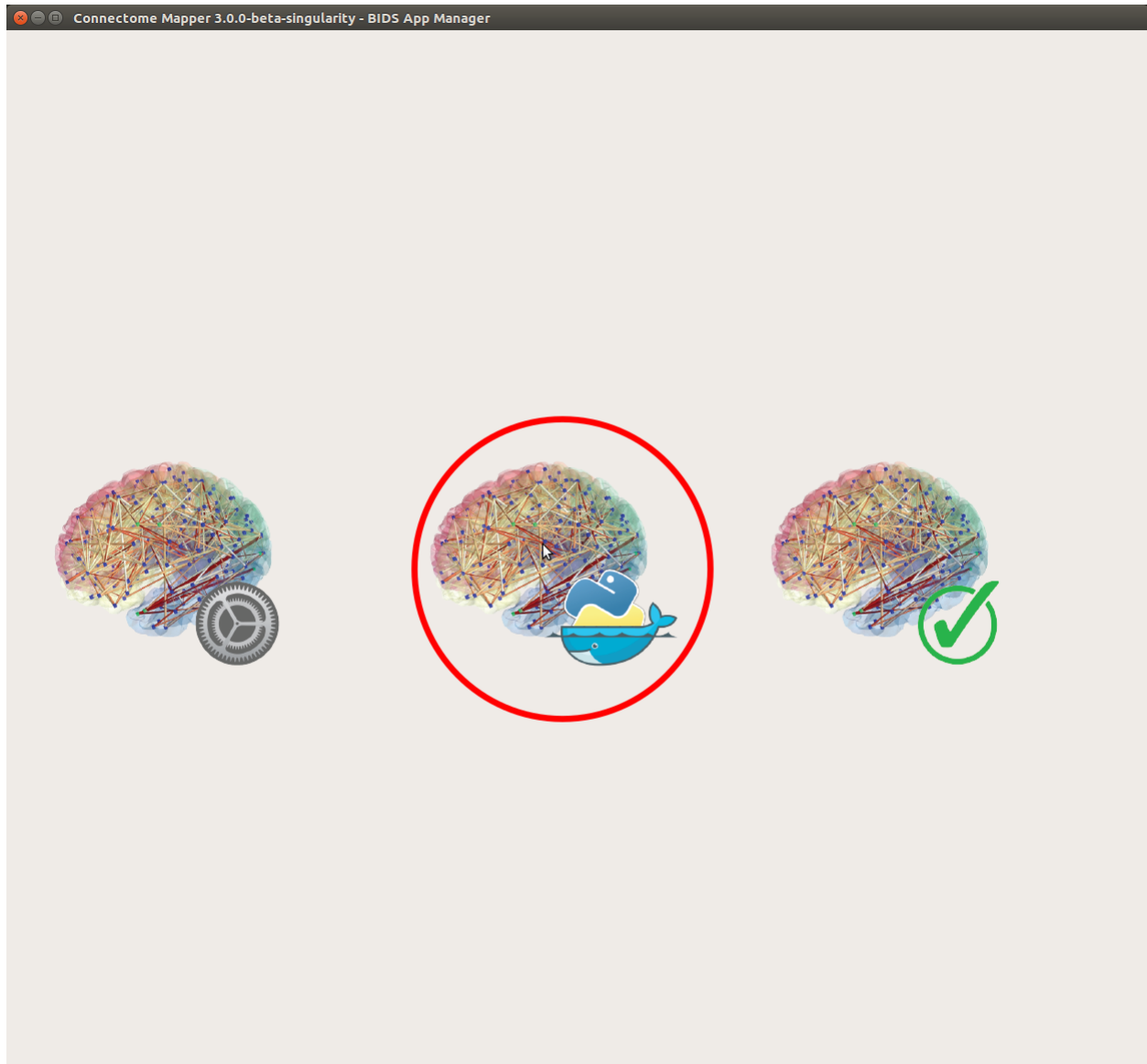
The Connectome Mapper processing relies on nipype. For each stage, a processing folder is created in `$Base_directory/derivatives/nipype/sub-<participant_label>/<pipeline_name>/<stage_name>`.

All intermediate steps for the processing are saved in the corresponding stage folders.

## **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 subject labels to be processed

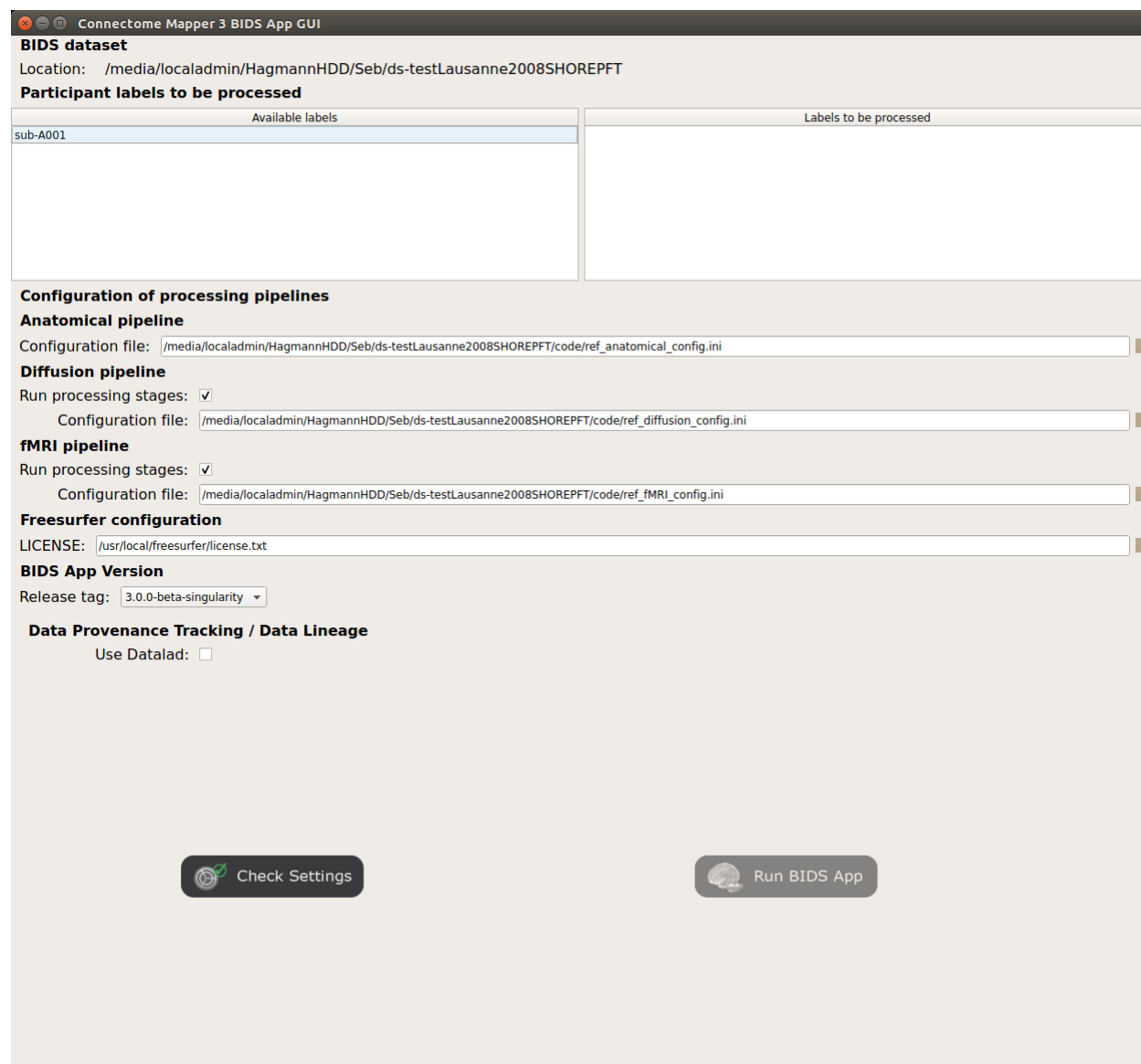


Fig. 7: Window of the Connectome Mapper BIDS App GUI



Connectome Mapper 3 BIDS App GUI

**BIDS dataset**  
Location: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT

**Participant labels to be processed**

Available labels	Labels to be processed
sub-A001	sub-A001

**Configuration of processing pipelines**

**Anatomical pipeline**  
Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_anatomical\_config.ini



**Diffusion pipeline**  
Run processing stages: ☒  
Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_diffusion\_config.ini

**fMRI pipeline**  
Run processing stages: ☒  
Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_fmri\_config.ini

**Freesurfer configuration**  
LICENSE: /usr/local/freesurfer/license.txt

**BIDS App Version**  
Release tag: 3.0.0-beta-singularity

**Data Provenance Tracking / Data Lineage**  
Use Datalad: ☐

- Check/Uncheck the pipelines to be performed

Connectome Mapper 3 BIDS App GUI

**BIDS dataset**  
Location: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT

**Participant labels to be processed**

Available labels	Labels to be processed
sub-A001	sub-A001

**Configuration of processing pipelines**

**Anatomical pipeline**  
Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_anatomical\_config.ini



**Diffusion pipeline**  
Run processing stages: ☒  
Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_diffusion\_config.ini

**fMRI pipeline**  
Run processing stages: ☐

**Freesurfer configuration**  
LICENSE: /usr/local/freesurfer/license.txt

**BIDS App Version**  
Release tag: 3.0.0-beta-singularity

**Data Provenance Tracking / Data Lineage**  
Use Datalad: ☐

- Specify your Freesurfer license

**Connectome Mapper 3 BIDS App GUI**

**BIDS dataset**  
 Location: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT

**Participant labels to be processed**

Available labels	Labels to be processed
sub-A001	sub-A001

**Configuration of processing pipelines**

**Anatomical pipeline**  
 Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_anatomical\_config.ini

**Diffusion pipeline**  
 Run processing stages: ☒  
 Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_diffusion\_config.ini

**fMRI pipeline**  
 Run processing stages: ☐

**Freesurfer configuration**  
 LICENSE: /usr/local/freesurfer/license.txt

**BIDS App Version**  
 Release tag: 3.0.0-beta-singularity

**Data Provenance Tracking / Data Lineage**  
 Use Datalad: ☐

Check Settings Run BIDS App

**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.

Connectome Mapper 3 BIDS App GUI

**BIDS dataset**  
Location: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT

**Participant labels to be processed**

Available labels	Labels to be processed
sub-A001	sub-A001

**Configuration of processing pipelines**

**Anatomical pipeline**  
Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_anatomical\_config.ini


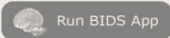
**Diffusion pipeline**  
Run processing stages: ☒  
Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_diffusion\_config.ini

**fMRI pipeline**  
Run processing stages: ☐

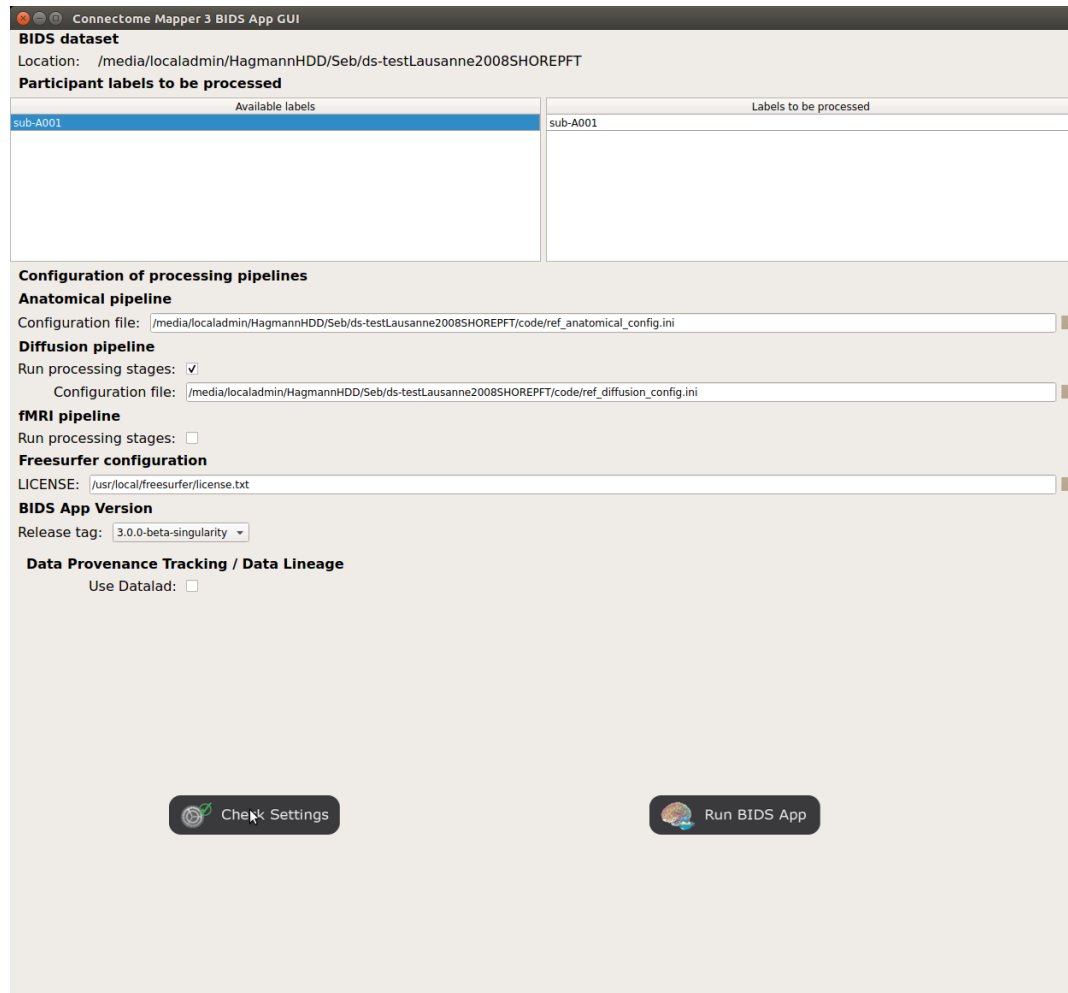
**Freesurfer configuration**  
LICENSE: /usr/local/freesurfer/license.txt

**BIDS App Version**  
Release tag: 3.0.0-beta-singularity

**Data Provenance Tracking / Data Lineage**  
Use Datalad: ☐

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



The screenshot shows the 'Connectome Mapper 3 BIDS App GUI'. At the top, it displays the 'BIDS dataset' location as '/media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT'. Below this, a section titled 'Participant labels to be processed' contains two tables. The 'Available labels' table has one entry, 'sub-A001', which is highlighted in blue. The 'Labels to be processed' table also has one entry, 'sub-A001'. The 'Configuration of processing pipelines' section includes three sub-sections: 'Anatomical pipeline' with a configuration file path, 'Diffusion pipeline' with a checked 'Run processing stages' checkbox and a configuration file path, and 'fMRI pipeline' with an unchecked 'Run processing stages' checkbox. The 'Freesurfer configuration' section shows a license file path. The 'BIDS App Version' section has a 'Release tag' dropdown set to '3.0.0-beta-singularity'. The 'Data Provenance Tracking / Data Lineage' section has an unchecked 'Use Datalad' checkbox. At the bottom, there are two buttons: 'Check Settings' and 'Run BIDS App'.

Connectome Mapper 3 BIDS App GUI

**BIDS dataset**  
Location: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT

**Participant labels to be processed**

Available labels	Labels to be processed
sub-A001	sub-A001

**Configuration of processing pipelines**

**Anatomical pipeline**  
Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_anatomical\_config.ini

**Diffusion pipeline**  
Run processing stages: ☒  
Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_diffusion\_config.ini

**fMRI pipeline**  
Run processing stages: ☐

**Freesurfer configuration**  
LICENSE: /usr/local/freesurfer/license.txt

**BIDS App Version**  
Release tag: 3.0.0-beta-singularity

**Data Provenance Tracking / Data Lineage**  
Use Datalad: ☐

Check Settings Run BIDS App

You are ready to launch the BIDS App run!

### Launch the BIDS App run

- Click on the Run BIDS App button to launch the BIDS App run

**Connectome Mapper 3 BIDS App GUI**

**BIDS dataset**  
 Location: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT  
**Participant labels to be processed**

Available labels	Labels to be processed
sub-A001	sub-A001

**Configuration of processing pipelines**

**Anatomical pipeline**  
 Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_anatomical\_config.ini

**Diffusion pipeline**  
 Run processing stages: ☒  
 Configuration file: /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/ref\_diffusion\_config.ini

**fMRI pipeline**  
 Run processing stages: ☐

**Freesurfer configuration**  
 LICENSE: /usr/local/freesurfer/license.txt

**BIDS App Version**  
 Release tag: 3.0.0-beta-singularity

**Data Provenance Tracking / Data Lineage**  
 Use Datalad: ☐

Check Settings Run BIDS App

- You can see the complete `docker run` command generated by the Connectome Mapper BIDS App GUI from the terminal

```
Start BIDS App
> Copy FreeSurfer license (BIDS App Manager)
... src : /usr/local/freesurfer/license.txt
... dst : /media/localadmin/HagmannHDD/Seb/ds-testLausanne2008SHOREPFT/code/
↳ license.txt
> Datalad available: True
*... Docker cmd 2 : ['docker', 'run', '-it', '--rm', '-v', '/media/localadmin/
↳ HagmannHDD/Seb/ds-testLausanne2008SHOREPFT:/tmp', '-u', '1000:1000',
↳ 'sebastientourbier/connectomemapper-bidsapp:3.0.0-beta-singularity', '/tmp',
↳ '/tmp/derivatives', 'participant', '--participant_label', 'A001', '--anat_
↳ pipeline_config', '/tmp/code/ref_anatomical_config.ini', '--dwi_pipeline_
↳ config', '/tmp/code/ref_diffusion_config.ini']*
> BIDS dataset: /tmp
> Subjects to analyze : ['A001']
> Copy FreeSurfer license (BIDS App)
> Sessions to analyze : ['ses-20150203160809']
> Process subject sub-A001 session ses-20150203160809
WARNING: rewriting config file /tmp/derivatives/sub-A001_ses-20150203160809_
↳ anatomical_config.ini
... Anatomical config created : /tmp/derivatives/sub-A001_ses-20150203160809_
↳ anatomical_config.ini
```

(continues on next page)

(continued from previous page)

```
WARNING: rewriting config file /tmp/derivatives/sub-A001_ses-20150203160809_
↳diffusion_config.ini
... Diffusion config created : /tmp/derivatives/sub-A001_ses-20150203160809_
↳diffusion_config.ini
... Running pipelines :
    - Anatomical MRI (segmentation and parcellation)
    - Diffusion MRI (structural connectivity matrices)
... cmd : connectomemapper3 /tmp /tmp/derivatives sub-A001 ses-20150203160809_
↳/tmp/derivatives/sub-A001_ses-20150203160809_anatomical_config.ini True /
↳tmp/derivatives/sub-A001_ses-20150203160809_diffusion_config.ini True
```

---

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

---

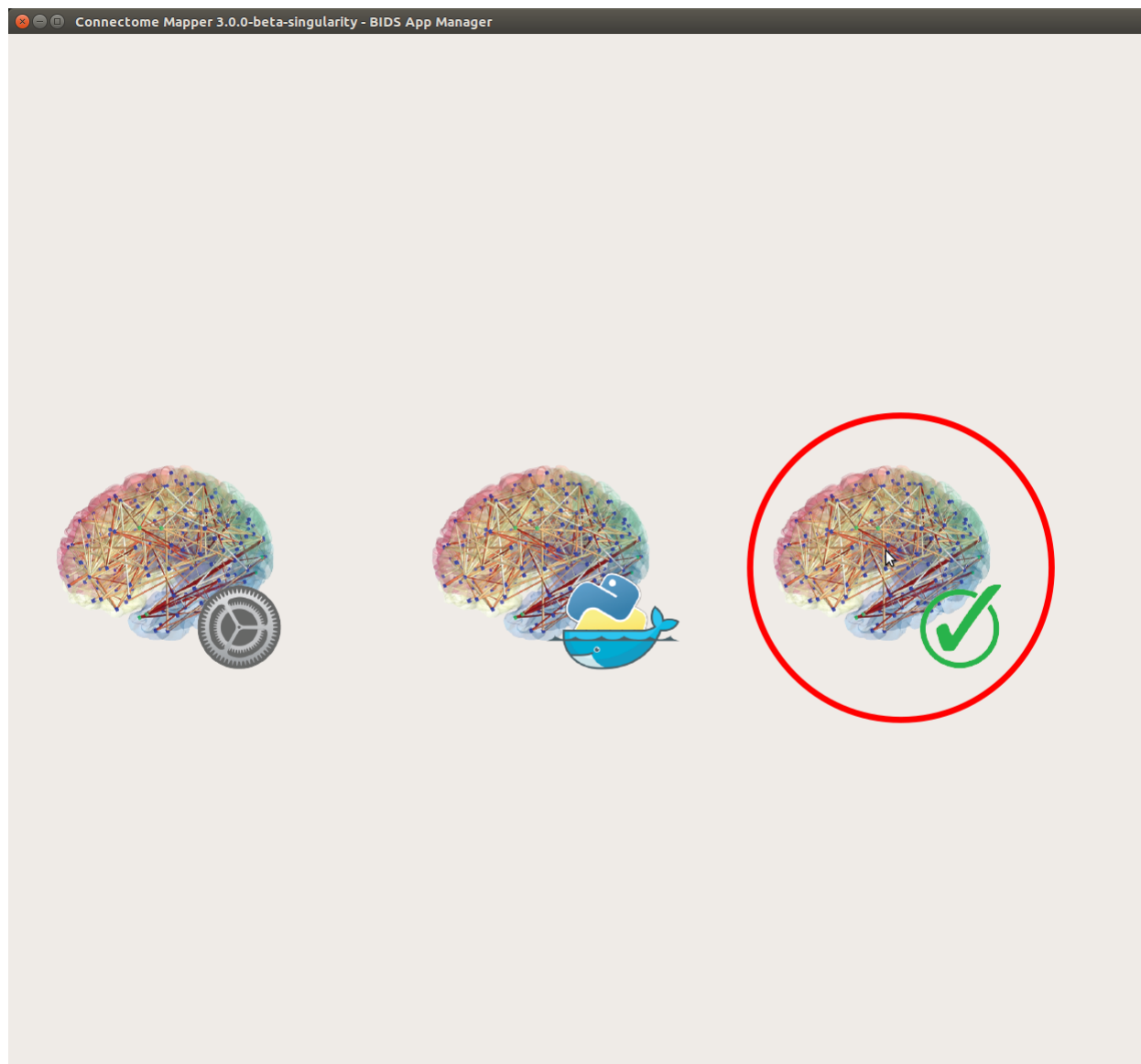
## Check progress

For each subject, the execution output of the pipelines are redirected to a log file, written as <bids\_dataset/derivatives>/cmp/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 window of the Connectome Mapper BIDS App Inspector 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):



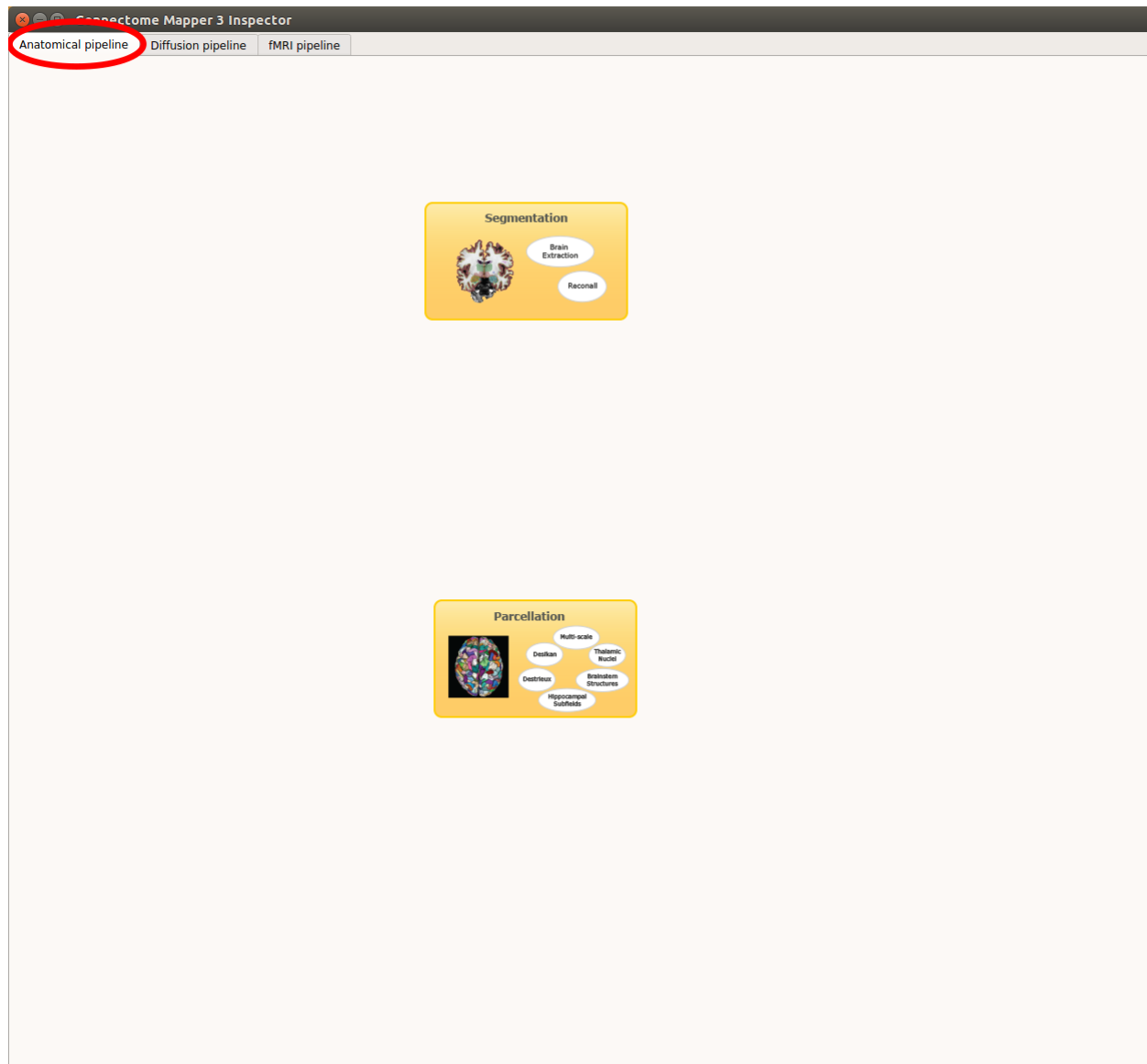
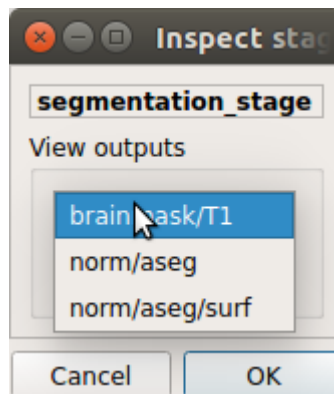


Fig. 8: Panel for configuration of anatomical pipeline stages

## Segmentation

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

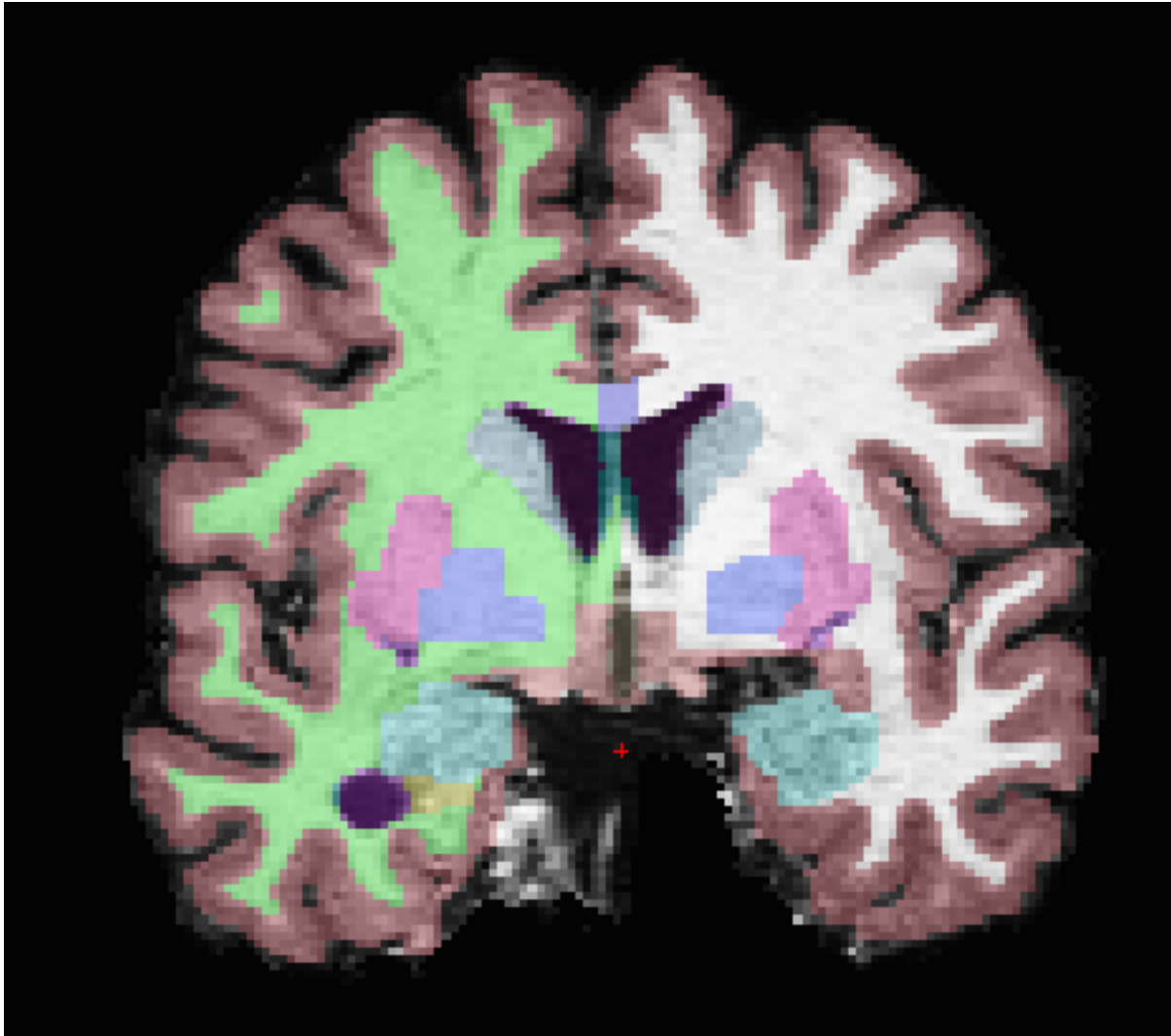


## Segmentation results

Surfaces extracted using Freesurfer.

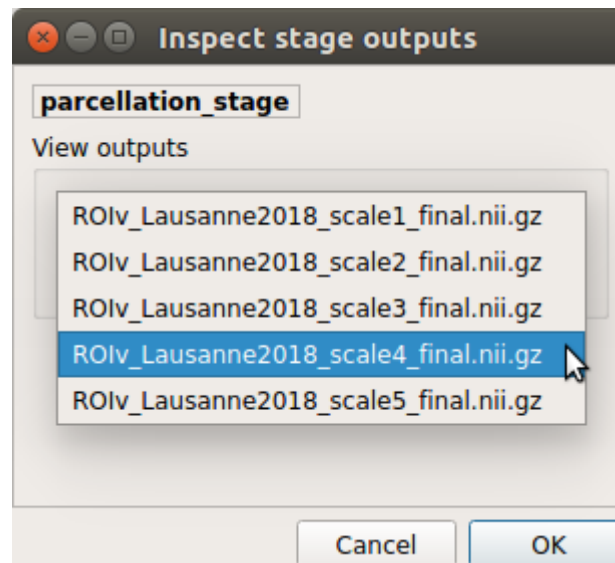


T1 segmented using Freesurfer.



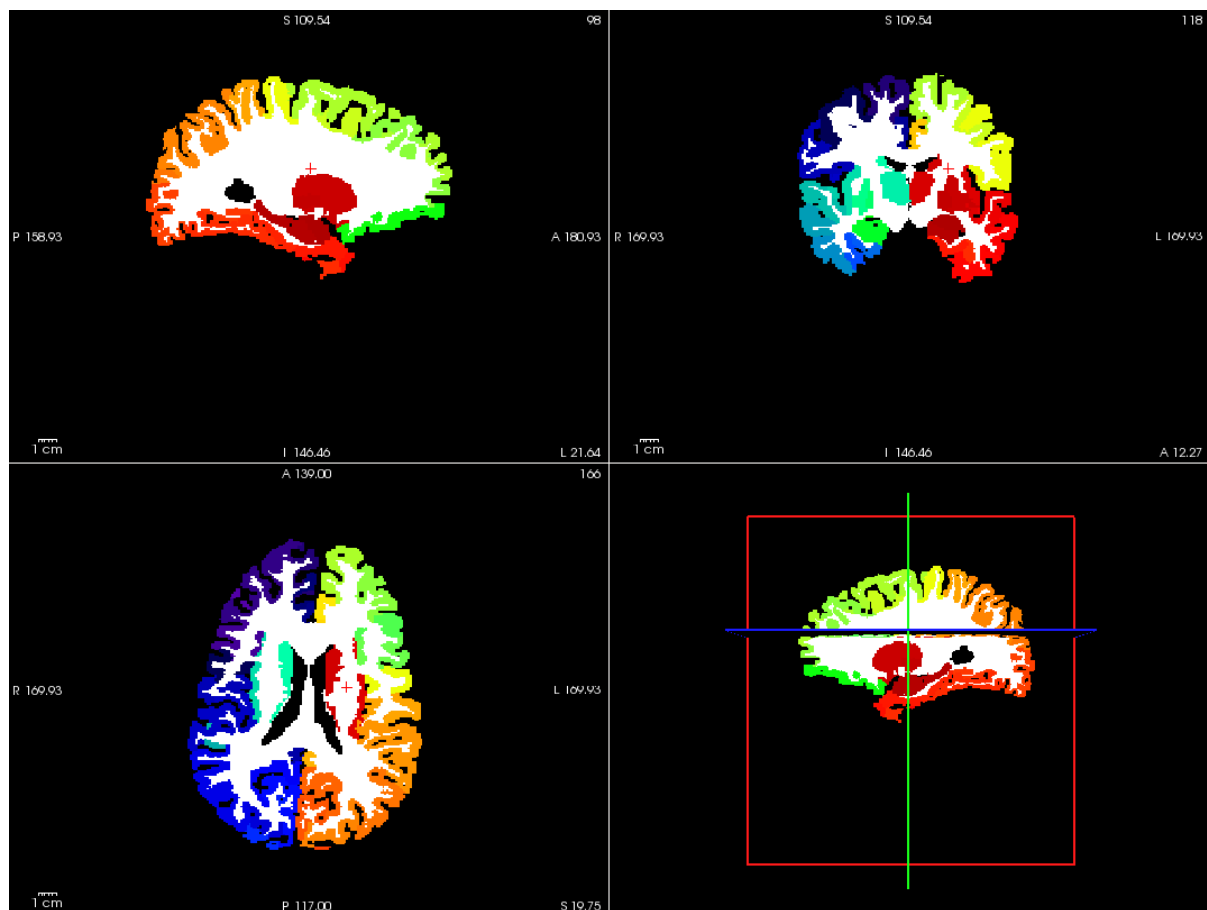
### Parcellation

- 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):

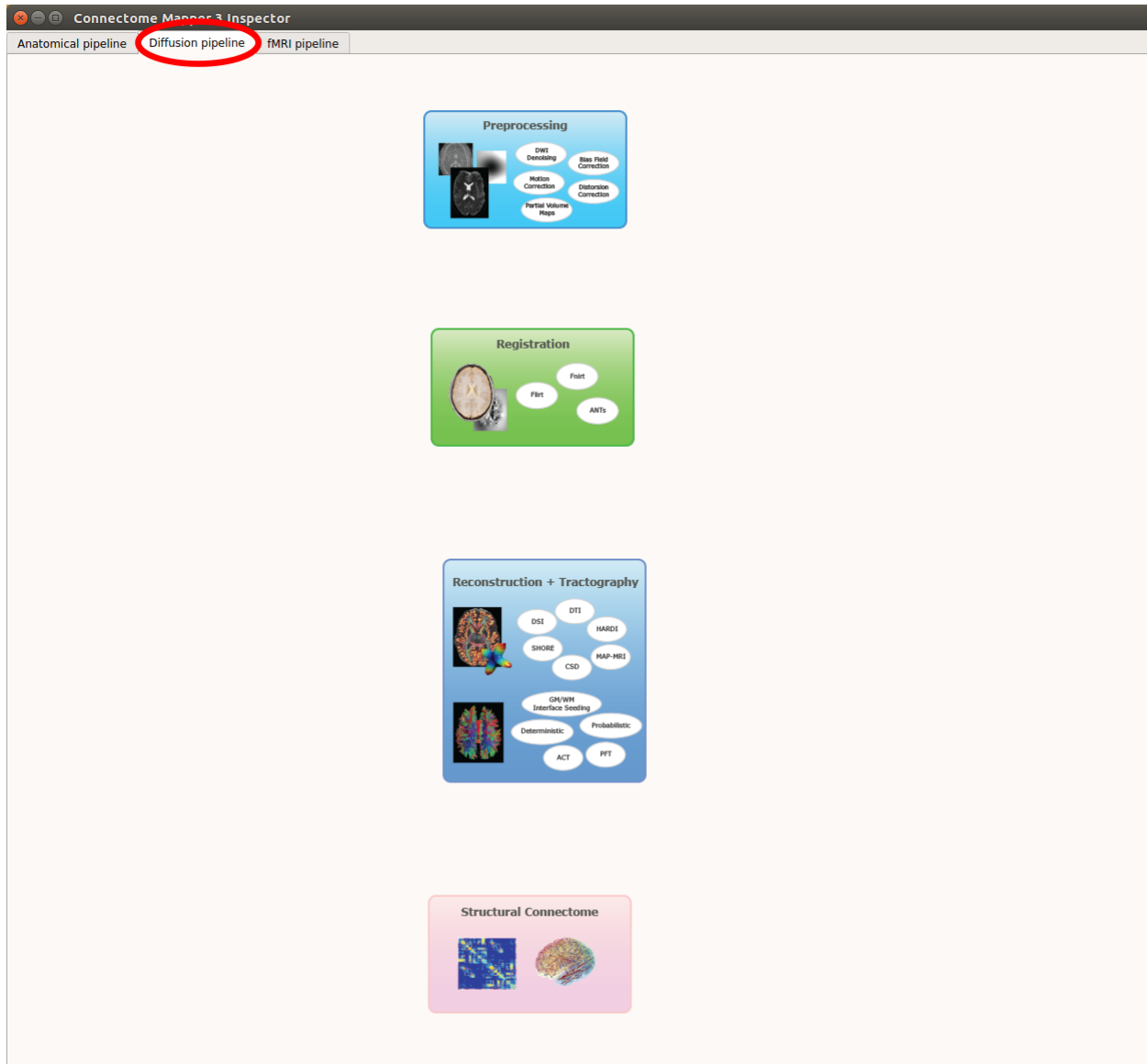
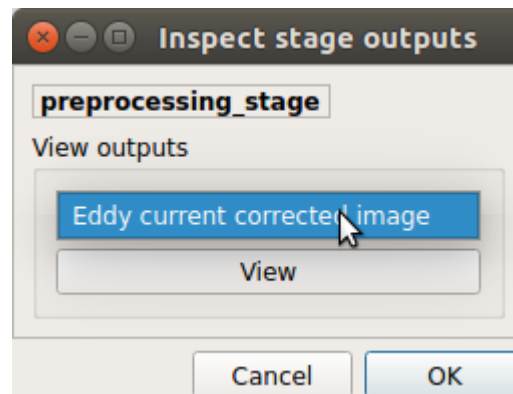


Fig. 9: Panel for configuration of diffusion pipeline stages

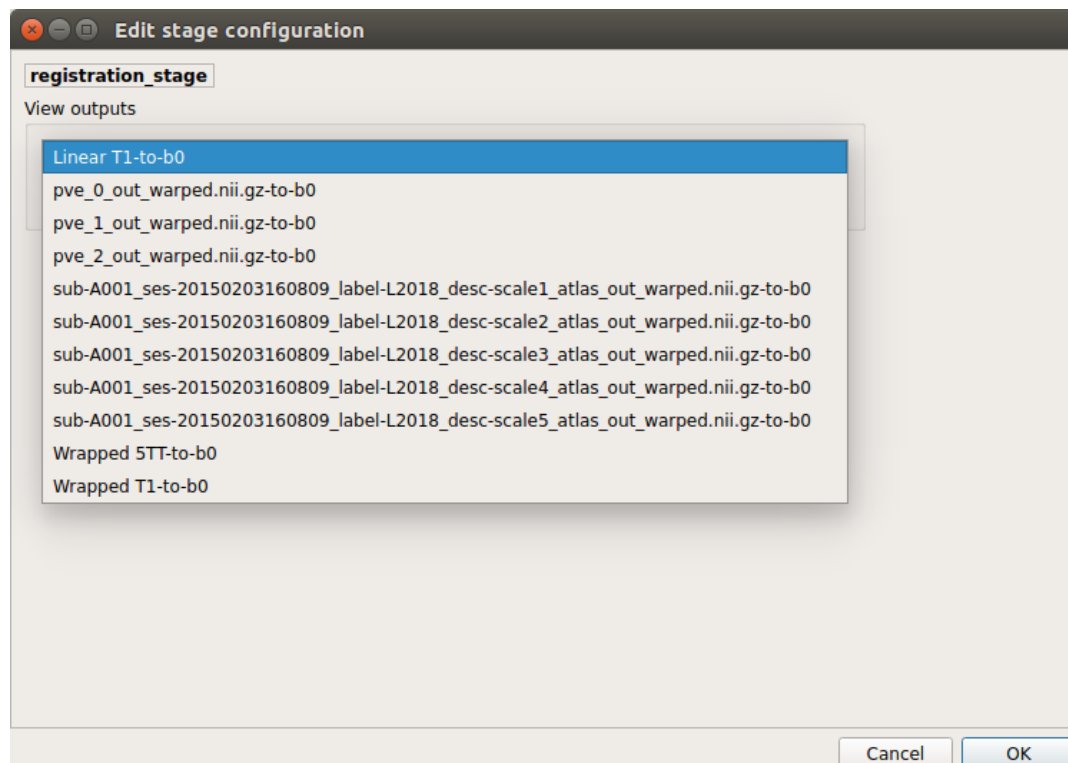
## Preprocessing

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



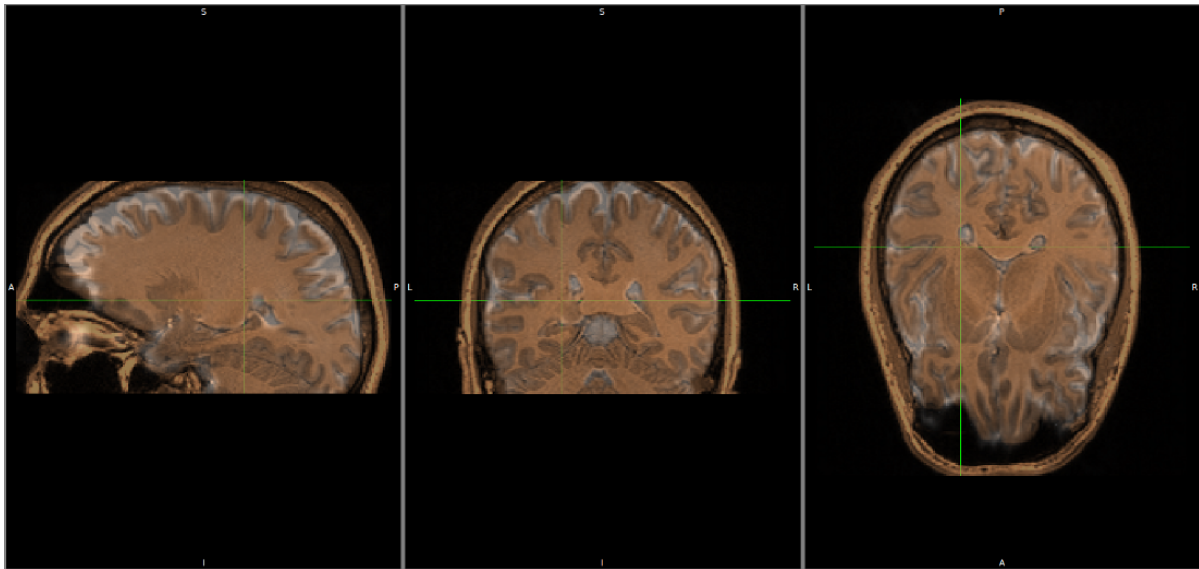
## Registration

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



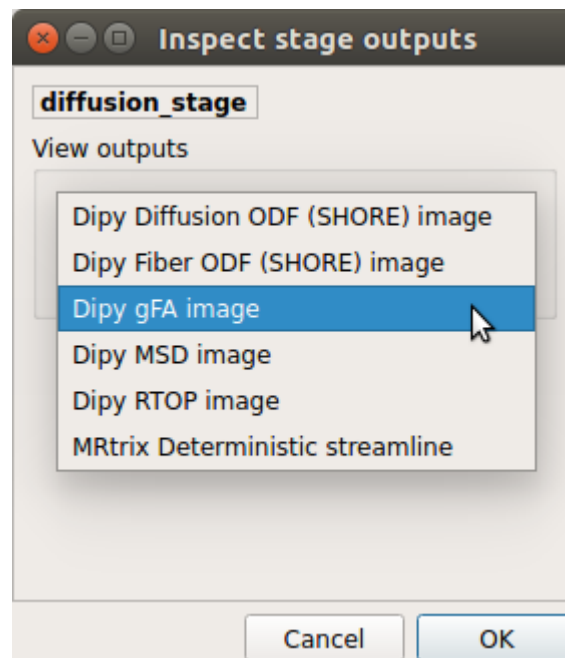
## Registration results

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



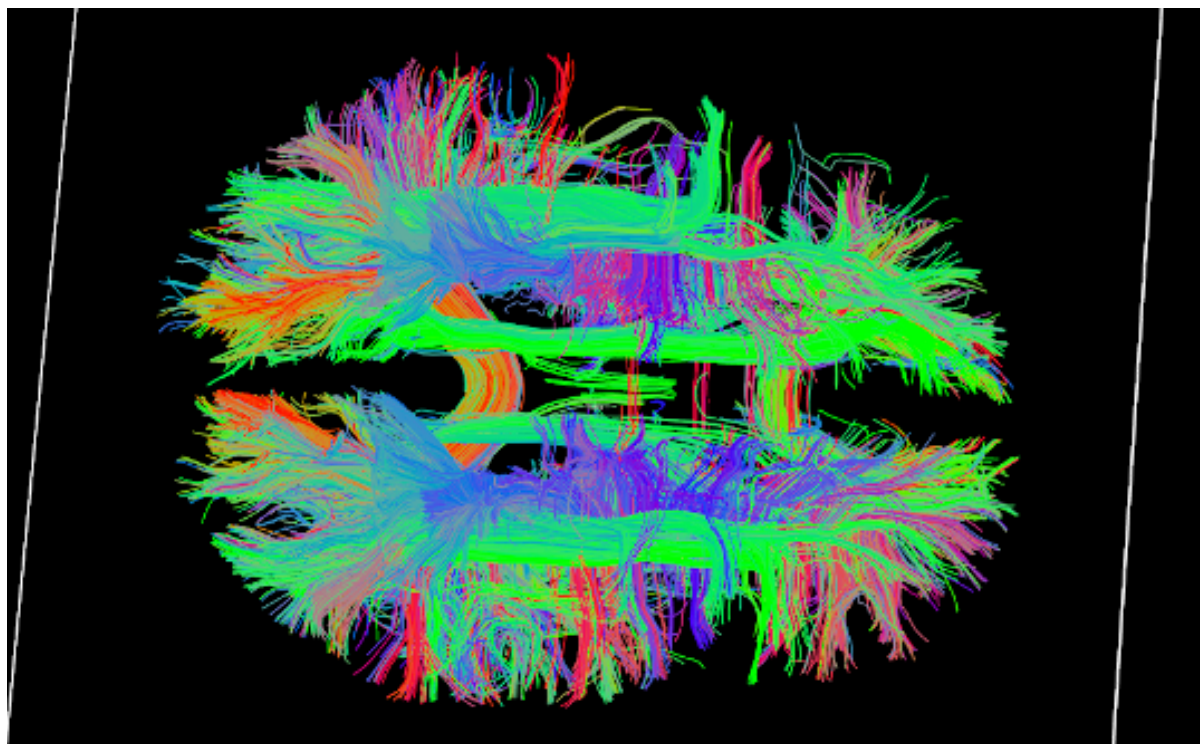
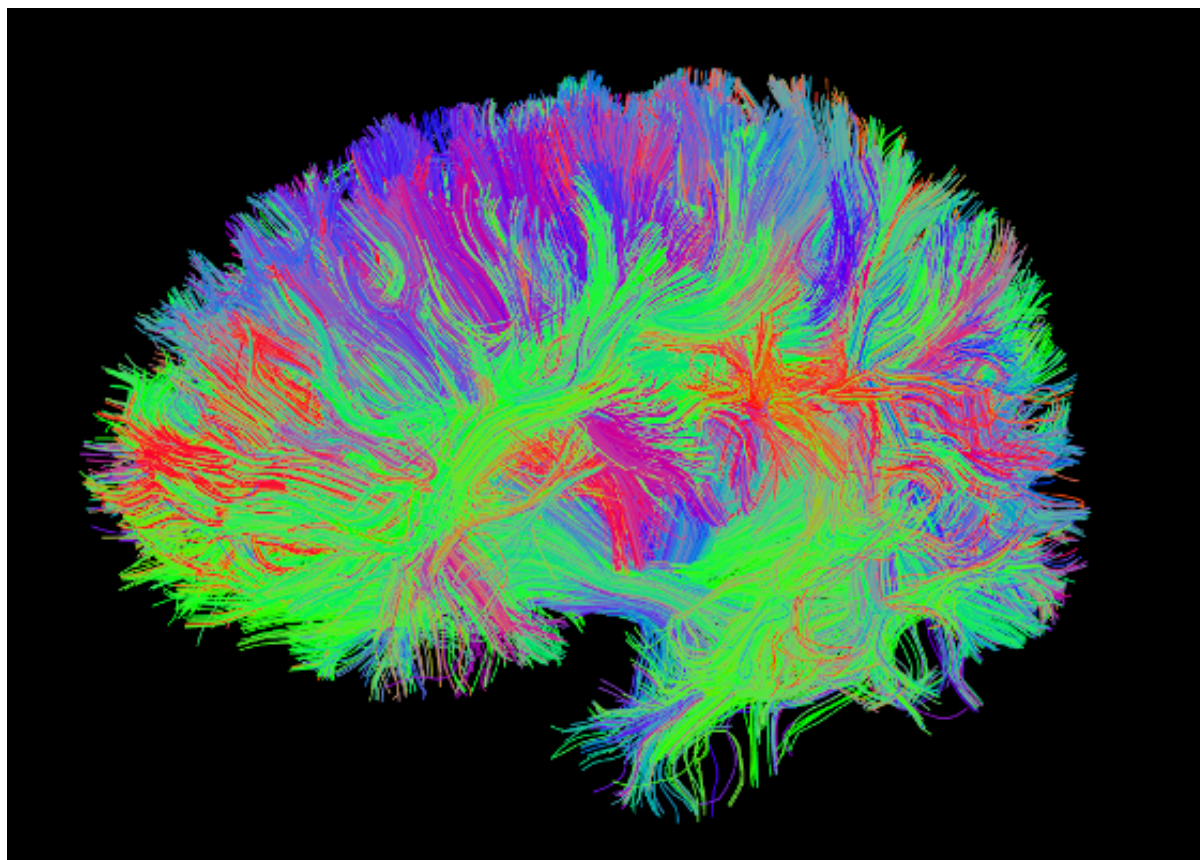
### Diffusion reconstruction and tractography

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



### Tractography results

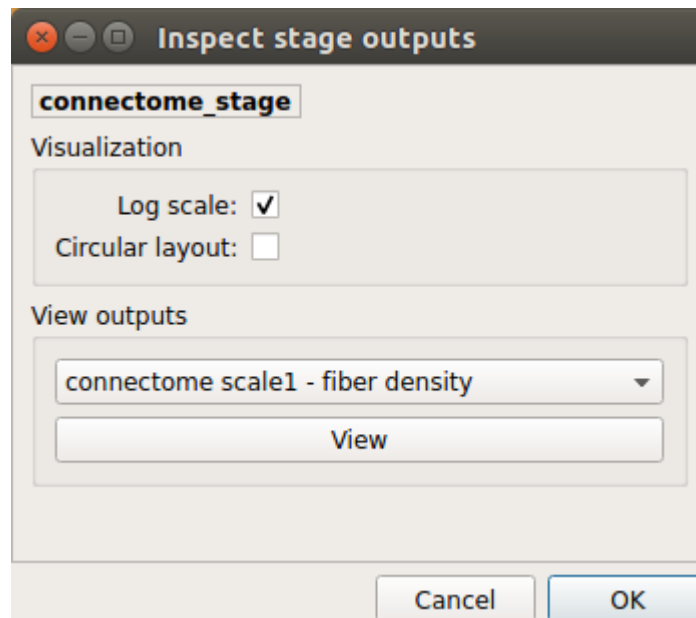
DSI Tractography results are displayed with TrackVis.





## Connectome

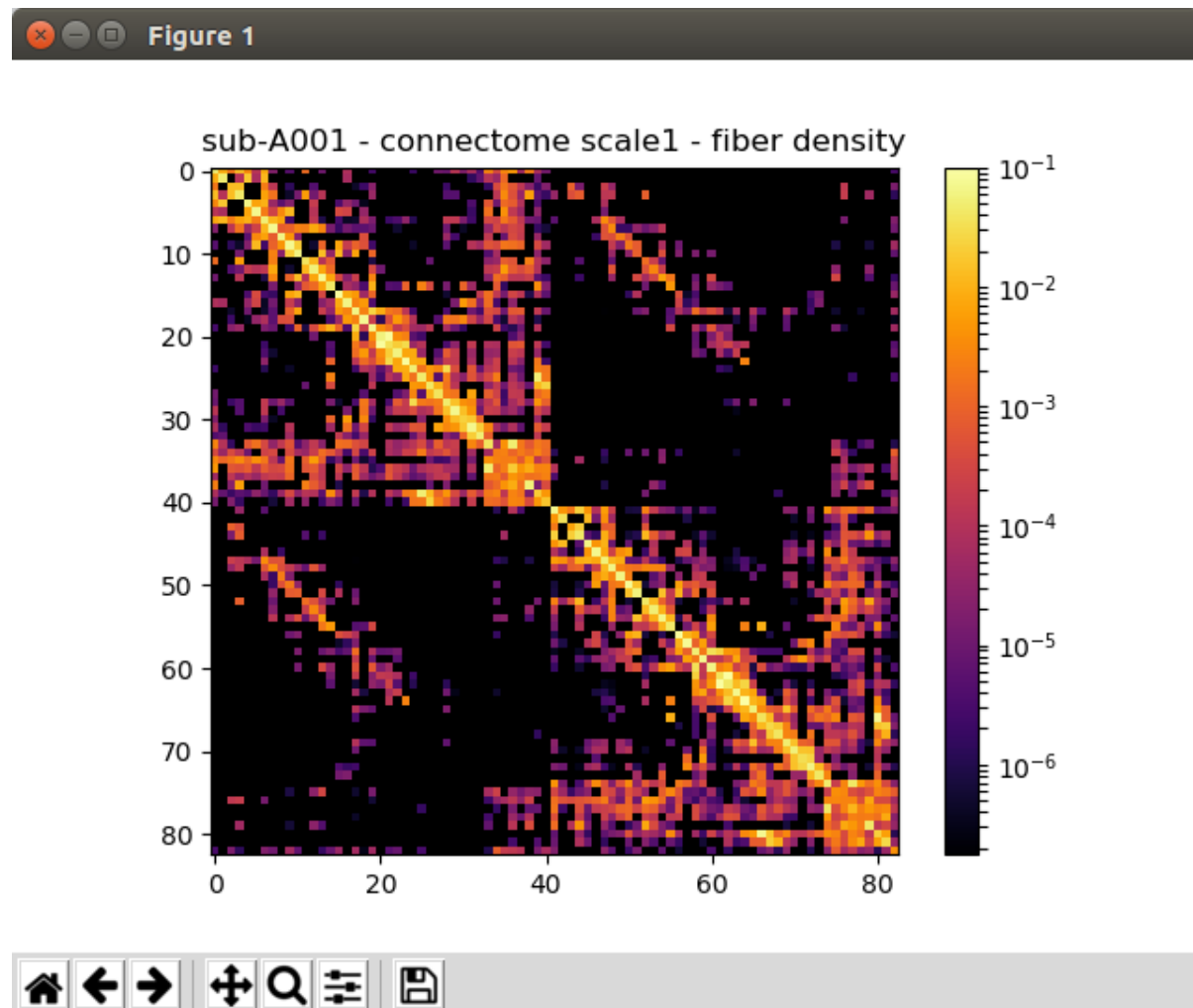
- Select the desired output from the list and click on view:



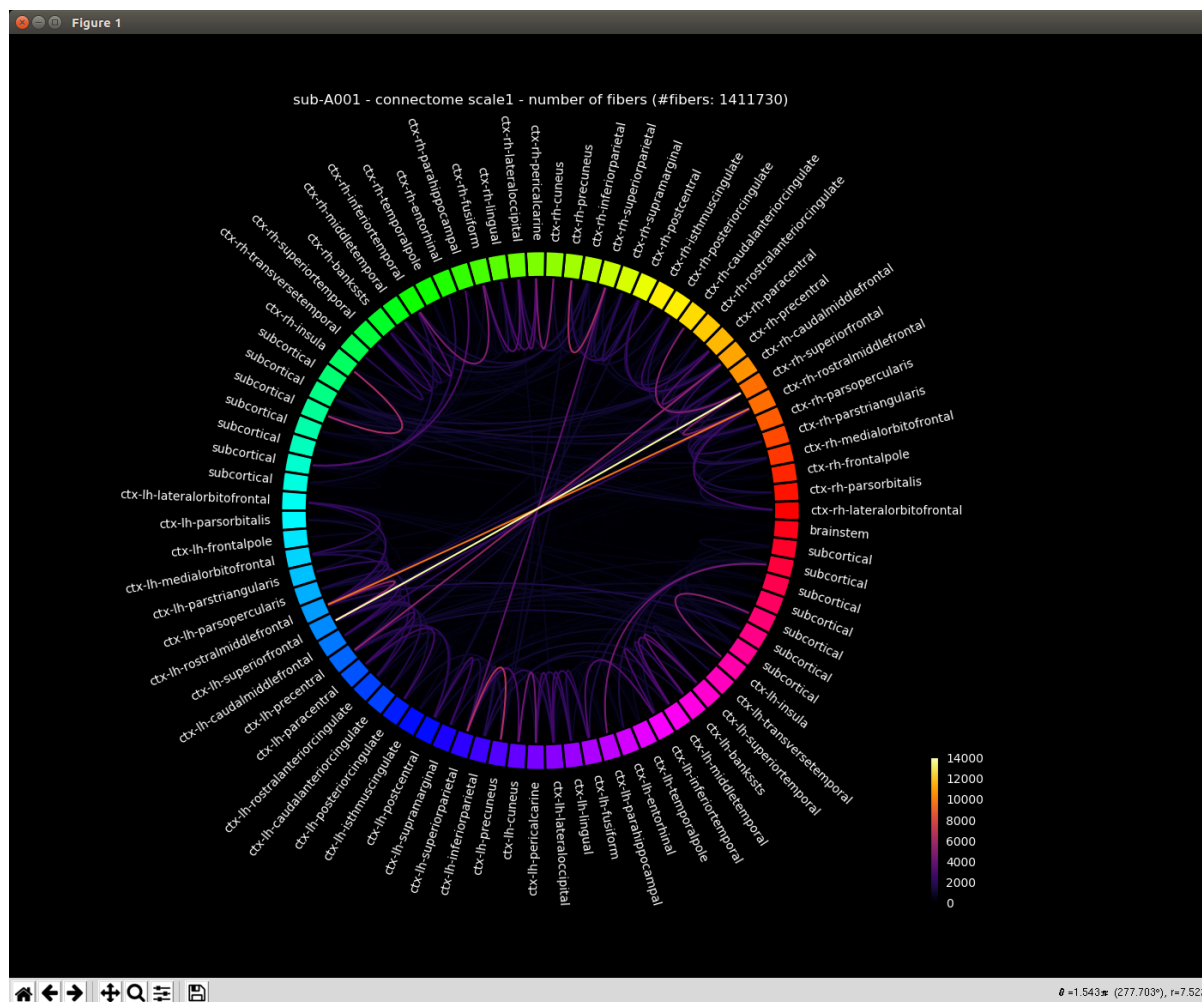
### Generated connection matrix

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):

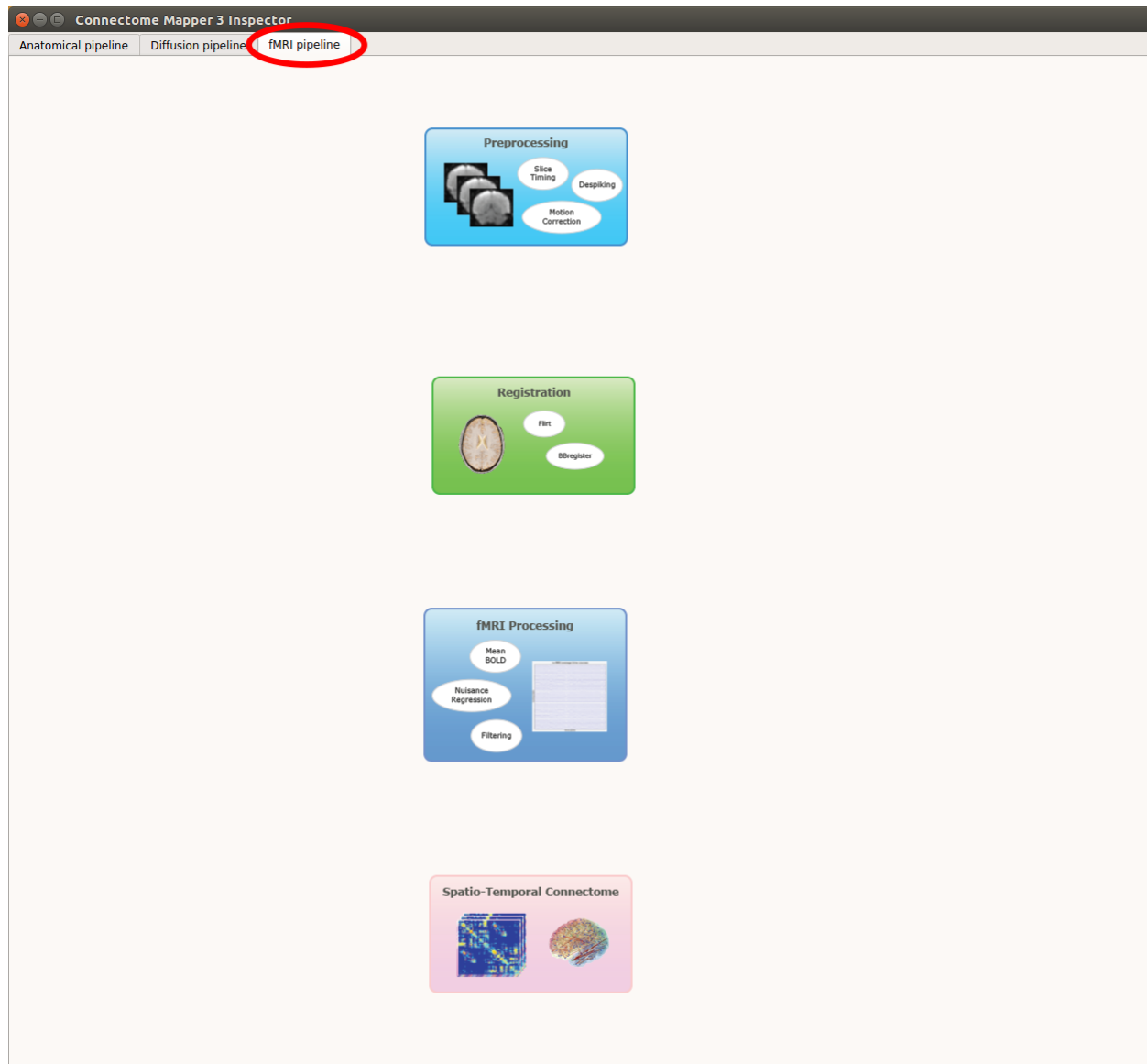
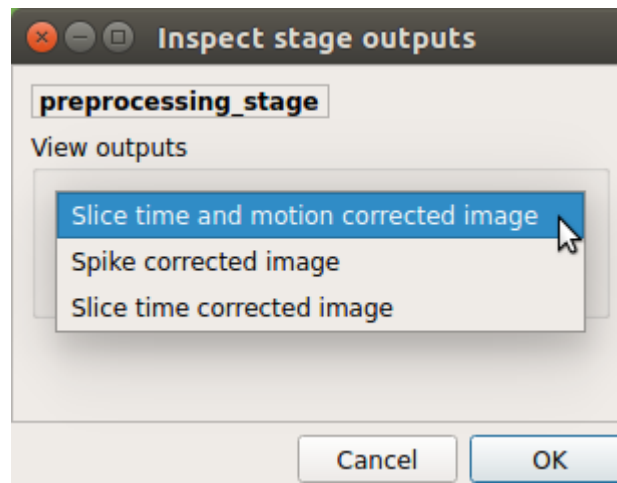


Fig. 10: Panel for configuration of fMRI pipeline stages

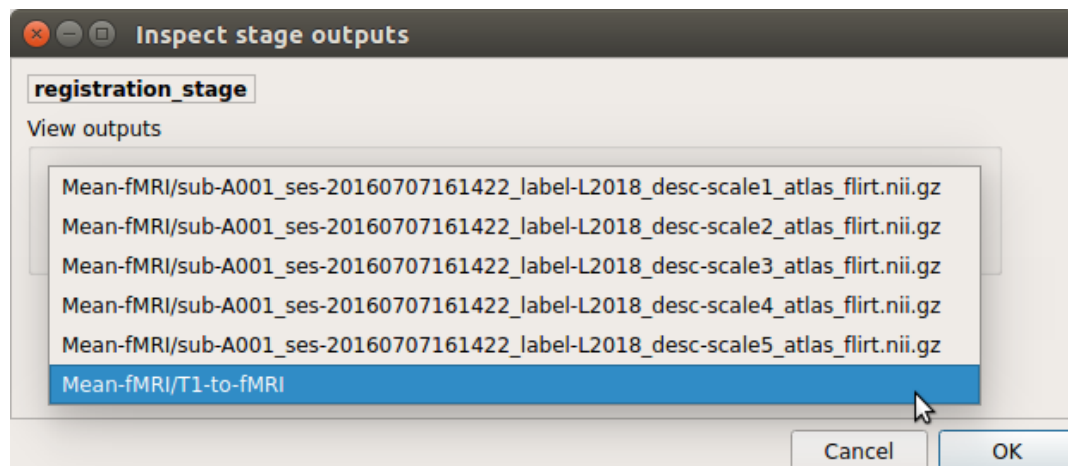
## Preprocessing

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



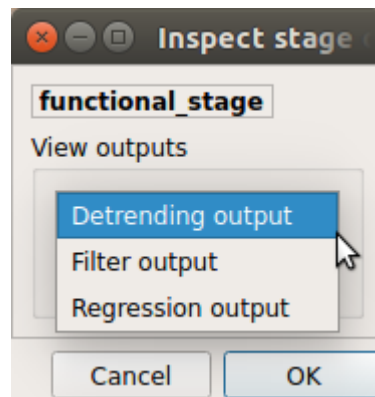
## Registration

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

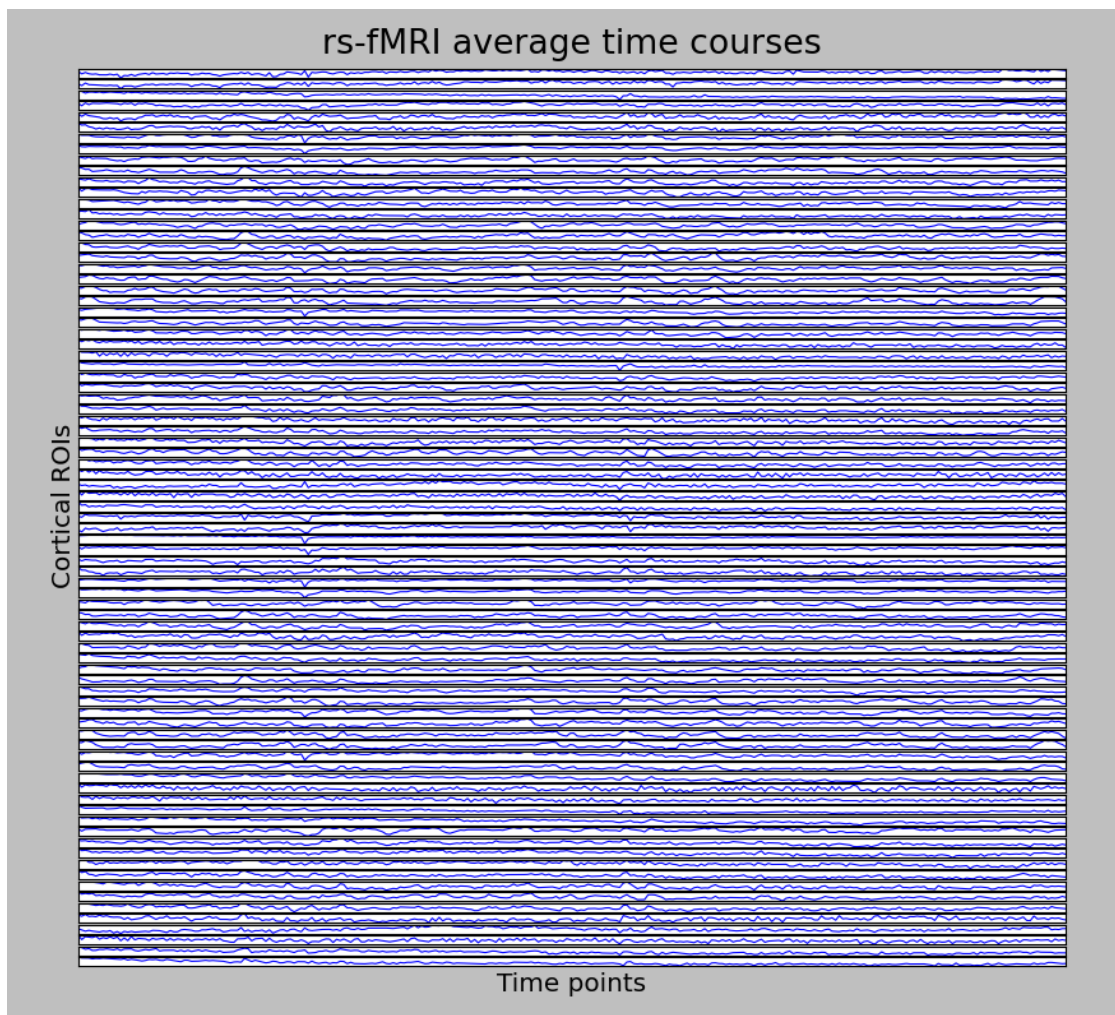


## fMRI processing

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

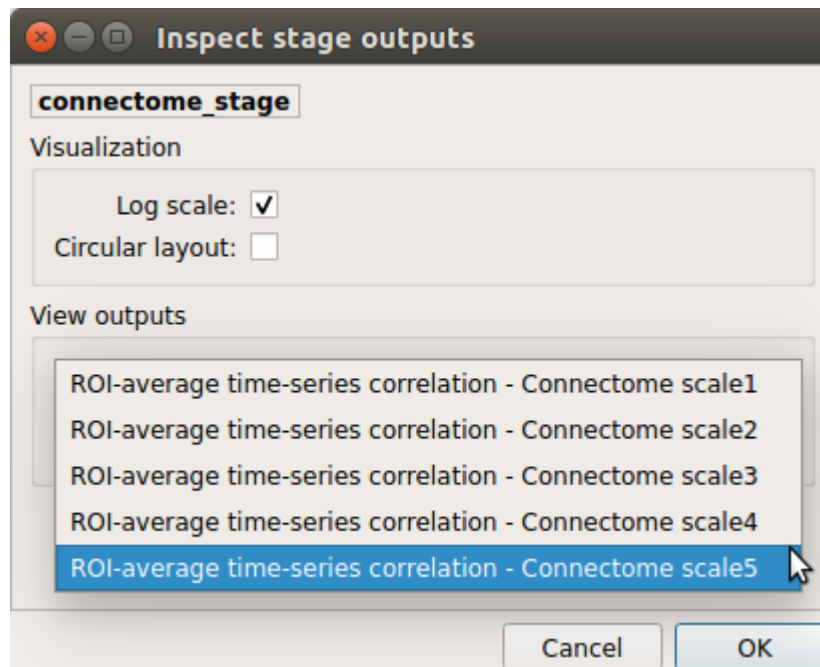


## ROI averaged time-series



## Connectome

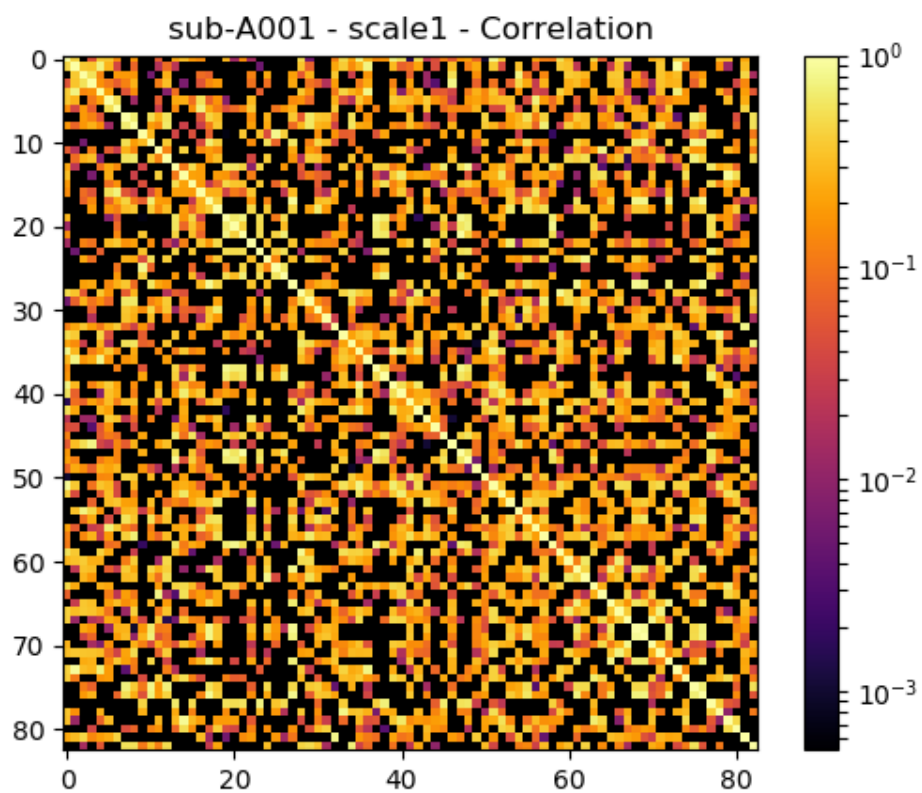
- Select the desired output from the list and click on view:



### Generated connection matrix

Displayed using a:

1. matrix layout with pyplot



2. circular layout with pyplot and MNE





## 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>\_label-L2018\_desc-<scale\_label>\_atlas.nii.gz

where <scale\_label> : scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale.

with the description of parcel labels and the updated FreeSurfer color lookup table:

- \* anat/sub-<subject\_label>\_label-L2018\_desc-<scale\_label>\_atlas.graphml

- \* anat/sub-<subject\_label>\_label-L2018\_desc-<scale\_label>\_atlas\_FreeSurferColor.txt

- Anatomical derivatives in the ``DWT`` 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\_label-<atlas\_label>\_desc-<scale\_label>\_atlas.nii.gz

where:

- <atlas\_label>:Desikan/L2008/L2018 is the parcellation scheme used

- <scale\_label> : scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale.

- The 5TT image used for Anatomically Constrained Tractography (ACT):
  - \* anat/sub-<subject\_label>\_space-DWI\_label-5TT\_probseg.nii.gz
- The partial volume maps for white matter (WM), gray matter (GM), and Cortical Spinal Fluid (CSF) used for Particle 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)
- <model\_label> is the type of tractography algorithm employed (DET for deterministic, PROB for probabilistic)

- The structural connectivity (SC) graphs:

- dwi/sub-<subject\_label>\_\_label-<atlas\_label>(\_desc-<scale\_label>)\_conndata-network<format>

where:

- <atlas\_label>: Desikan/L2008/L2018 is the parcellation scheme used
- <scale\_label>: scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale if applicable
- <format>: mat/gpickle/tsv/graphml is the preferred format employed 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-L2018\_desc-<scale\_label>\_timeseries.npy

```
- func/sub-<subject_label>_atlas-L2018_desc-<scale_label>_timeseries.
  mat
```

where <scale\_label> : scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale

- The functional connectivity (FC) graphs:

```
- func/sub-<subject_label>__label-<atlas_label>(_desc-<scale_label>)_conndata-netw
  <format>
```

where:

- <atlas\_label>: Desikan, L2008, L2018 is the parcellation scheme used
- <scale\_label>: scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale if applicable
- <format>: mat, gpickle, tsv, graphml is the preferred format employed to stored the graph

## 5.5.2 FreeSurfer Derivatives

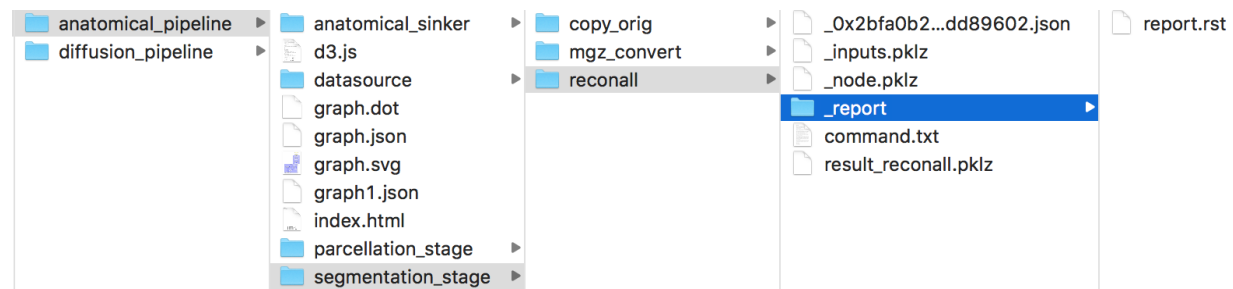
A FreeSurfer subjects directory is created in <bids\_dataset/derivatives>/freesurfer.

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

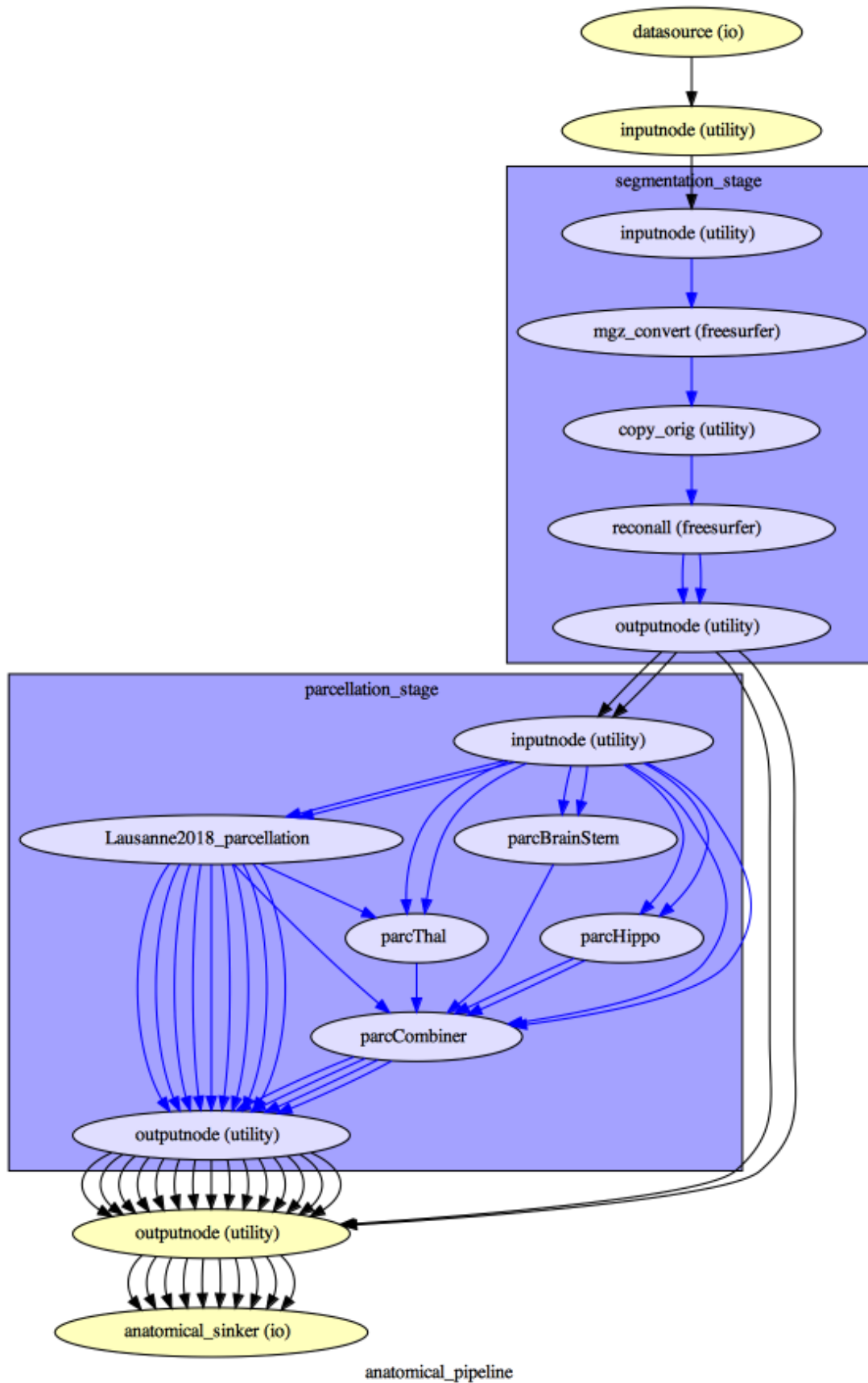
The fsaverage subject distributed with the running version of FreeSurfer is copied into this directory.

## 5.5.3 Nipype Workflow Derivatives

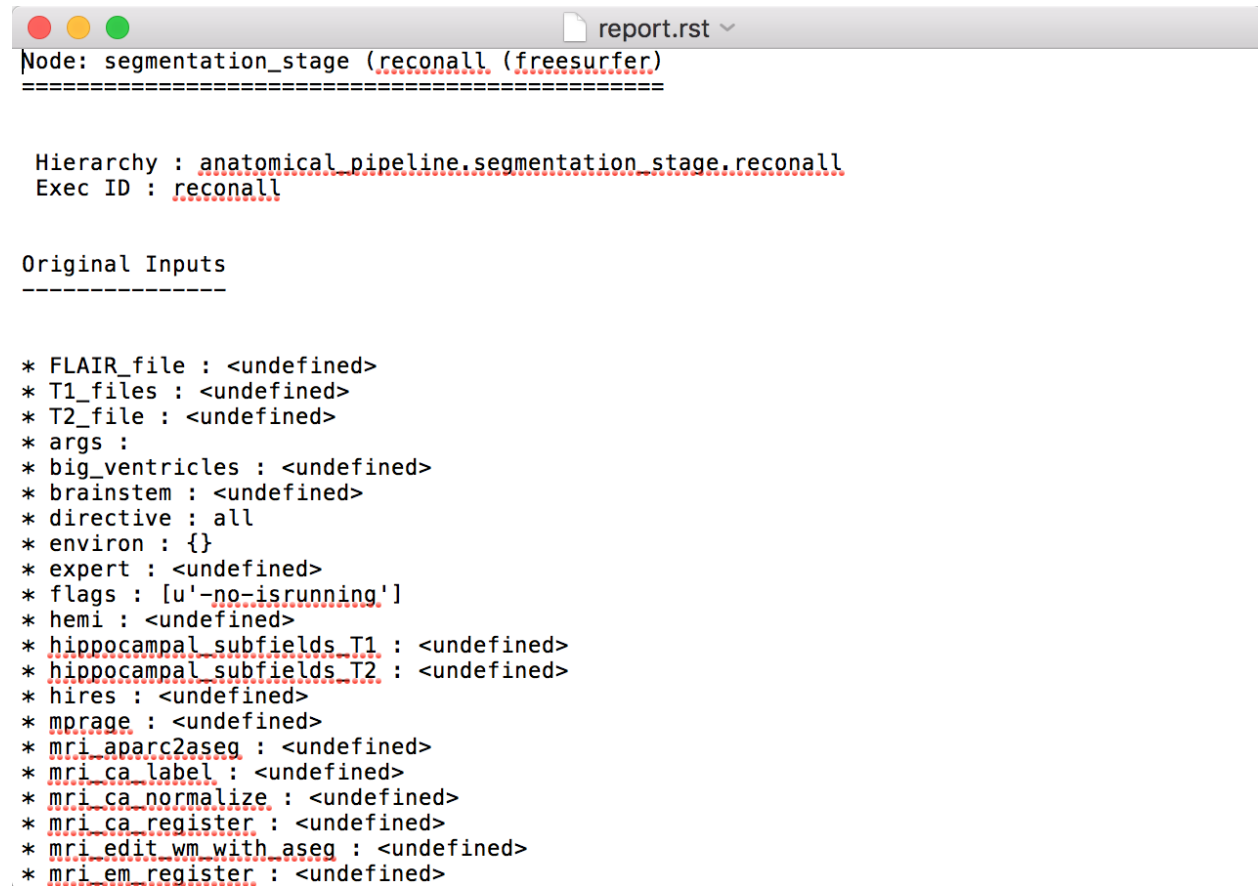
The execution of each Nipype workflow (pipeline) dedicated to the processing of one modality (i.e. anatomical/diffusion/fMRI) involves the creation of a number of intermediate outputs which are written to <bids\_dataset/derivatives>/nipype/sub-<subject\_label>/<anatomical/diffusion/fMRI>\_pipeline respectively:



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



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

Return the argparse parser of the BIDS App.

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

**Return type** object

## cmp.project module

Definition of classes and functions for handling non-GUI general events.

**class** cmp.project.**CMP\_Project\_Info**

Bases: traits.has\_traits.HasTraits

Class used to store all properties of a processing project.

**base\_directory**

BIDS dataset root directory

**Type** traits.Directory

**output\_directory**

Output directory

**Type** traits.Directory

**bids\_layout**

Instance of pybids BIDSLayout

**Type** bids.BIDSLayout

**subjects**

List of subjects in the dataset

**Type** traits.List

**subject**

Subject being processed in the form sub-XX

**subject\_sessions**

List of sessions for the subject being processed

**Type** traits.List

**subject\_session**

Session of the subject being processed in the form ses-YY

**Type** trait.Str

**diffusion\_imaging\_model**

Diffusion imaging model that can be 'DSI', 'DTI', or 'multi-shell'

**Type** traits.Str

**dmri\_bids\_acqs**

List diffusion imaging models extracted from acq-<label> filename part.

**Type** traits.List

**dmri\_bids\_acq**

Diffusion imaging model being processed

**anat\_runs**

List of run labels for T1w scans with multiple runs

**Type** traits.List

**anat\_run**

Run being processed for T1w scans with multiple runs

**Type** traits.Str



**dmri\_runs**

List of run labels for DWI scans with multiple runs

**Type** traits.List

**dmri\_run**

Run being processed for DWI scans with multiple runs

**Type** traits.Str

**fmri\_runs**

List of run labels for fMRI scans with multiple runs

**Type** traits.List

**fmri\_run**

Run being processed for fMRI scans with multiple runs

**Type** traits.Str

**parcellation\_scheme**

Parcellation scheme used (Default: 'Lausanne2008')

**Type** traits.Str

**atlas\_info**

Dictionary storing parcellation atlas information See `ParcellationStage` for more details

**Type** traits.Dict

**freesurfer\_subjects\_dir**

Freesurfer subjects directory

**Type** traits.Str

**freesurfer\_subject\_id**

Freesurfer subject ID

**Type** traits.Str

**t1\_available**

True if T1w scans were found (Default: False)

**Type** Bool

**dmri\_available**

True if DWI scans were found (Default: False)

**Type** Bool

**fmri\_available**

True if fMRI scans were found (Default: False)

**Type** Bool

**anat\_config\_error\_msg**

Error message for the anatomical pipeline configuration file

**Type** traits.Str

**anat\_config\_to\_load**

Path to a configuration file for the anatomical pipeline

**Type** traits.Str

**anat\_available\_config**

List of configuration files for the anatomical pipeline

**Type** traits.List

**anat\_stage\_names**  
List of anatomical pipeline stage names

**Type** traits.List

**anat\_custom\_last\_stage**  
Custom last anatomical pipeline stage to be processed

**Type** traits.Str

**dmri\_config\_error\_msg**  
Error message for the diffusion pipeline configuration file

**Type** traits.Str

**dmri\_config\_to\_load**  
Path to a configuration file for the diffusion pipeline

**Type** traits.Str

**dmri\_available\_config**  
List of configuration files for the anatomical pipeline

**Type** traits.List

**dmri\_stage\_names**  
List of diffusion pipeline stage names

**Type** traits.List

**dmri\_custom\_last\_stage**  
Custom last diffusion pipeline stage to be processed

**Type** traits.Str

**fmri\_config\_error\_msg**  
Error message for the fMRI pipeline configuration file

**Type** traits.Str

**fmri\_config\_to\_load**  
Path to a configuration file for the fMRI pipeline

**Type** traits.Str

**fmri\_available\_config**  
List of configuration files for the fMRI pipeline

**Type** traits.List

**fmri\_stage\_names**  
List of fMRI pipeline stage names

**Type** traits.List

**fmri\_custom\_last\_stage**  
Custom last fMRI pipeline stage to be processed

**Type** traits.Str

**number\_of\_cores**  
Number of cores used by Nipype workflow execution engine to distribute independent processing nodes  
(Must be in the range of your local resources)

**Type** int

`cmp.project.init_anat_project(project_info, is_new_project, debug=False)`

Initialize the anatomical processing pipeline.

#### Parameters

- **project\_info** (`cmp.project.CMP_Project_Info`) – Instance of `cmp.project.CMP_Project_Info` object
- **is\_new\_project** (*bool*) – Specify if it corresponds or not to a new project. If `True`, it will create initial pipeline configuration files.
- **debug** (*bool*) – If `True`, display extra prints to support debugging

**Returns** `anat_pipeline` – `AnatomicalPipeline` object instance

**Return type** `Instance(cmp.pipelines.anatomical.anatomical.AnatomicalPipeline)`

`cmp.project.init_dmri_project(project_info, bids_layout, is_new_project, gui=True, debug=False)`

Initialize the diffusion processing pipeline.

#### Parameters

- **project\_info** (`cmp.project.CMP_Project_Info`) – Instance of `cmp.project.CMP_Project_Info` object
- **bids\_layout** (`bids.BIDSLayout`) – Instance of `BIDSLayout` object
- **is\_new\_project** (*bool*) – Specify if it corresponds or not to a new project. If `True`, it will create initial pipeline configuration files.
- **gui** (*bool*) – Might be obsolete and removed in future versions
- **debug** (*bool*) – If `True`, display extra prints to support debugging

**Returns** `dmri_pipeline` – `DiffusionPipeline` object instance

**Return type** `Instance(cmp.pipelines.diffusion.diffusion.DiffusionPipeline)`

`cmp.project.init_fmri_project(project_info, bids_layout, is_new_project, gui=True, debug=False)`

Initialize the fMRI processing pipeline.

#### Parameters

- **project\_info** (`cmp.project.CMP_Project_Info`) – Instance of `cmp.project.CMP_Project_Info` object
- **bids\_layout** (`bids.BIDSLayout`) – Instance of `BIDSLayout` object
- **is\_new\_project** (*bool*) – Specify if it corresponds or not to a new project. If `True`, it will create initial pipeline configuration files.
- **gui** (*bool*) – Might be obsolete and removed in future versions
- **debug** (*bool*) – If `True`, display extra prints to support debugging

**Returns** `fmri_pipeline` – `fMRIPipeline` object instance

**Return type** `Instance(cmp.pipelines.functional.fMRI.fMRIPipeline)`

`cmp.project.refresh_folder(bids_directory, derivatives_directory, subject, input_folders, session=None)`

Creates (if needed) the folder hierarchy.

#### Parameters

- **bids\_directory** (*os.path*) – BIDS dataset root directory

- **derivatives\_directory** (*os.path*) – Output (derivatives) directory
- **subject** (*string*) – BIDS subject label (sub-XX)
- **input\_folders** (*List of string*) – List of folder to be created in derivatives\_directory/'cmp'/subject
- **session** (*string*) – BIDS session label (ses-YY)

```
cmp.project.run_individual(bids_dir, output_dir, participant_label, session_label,  
                          anat_pipeline_config, dwi_pipeline_config, func_pipeline_config,  
                          number_of_threads=1)
```

Function that creates the processing pipeline for complete coverage.

#### Parameters

- **bids\_dir** (*string*) – BIDS dataset root directory
- **output\_dir** (*string*) – Output (derivatives) directory
- **participant\_label** (*string*) – BIDS participant / subject label (sub-XX)
- **session\_label** (*string*) – BIDS session label (ses-XX)
- **anat\_pipeline\_config** (*string*) – Path to anatomical pipeline configuration file
- **dwi\_pipeline\_config** (*string*) – Path to diffusion pipeline configuration file
- **func\_pipeline\_config** (*string*) – Path to fMRI pipeline configuration file
- **number\_of\_threads** (*int*) – Number of threads used by programs relying on the OpenMP library

```
cmp.project.update_anat_last_processed(project_info, pipeline)
```

Update last processing information of a *AnatomicalPipeline*.

#### Parameters

- **project\_info** (`cmp.project.CMP_Project_Info`) – Instance of *CMP\_Project\_Info* object
- **pipeline** (`cmp.pipelines.anatomical.anatomical.AnatomicalPipeline`) – Instance of *AnatomicalPipeline* object

```
cmp.project.update_dmri_last_processed(project_info, pipeline)
```

Update last processing information of an *DiffusionPipeline*.

#### Parameters

- **project\_info** (`cmp.project.CMP_Project_Info`) – Instance of *CMP\_Project\_Info* object
- **pipeline** (`cmp.pipelines.diffusion.diffusion.DiffusionPipeline`) – Instance of *DiffusionPipeline* object

```
cmp.project.update_fmri_last_processed(project_info, pipeline)
```

Update last processing information of an *fMRIPipeline*.

#### Parameters

- **project\_info** (`cmp.project.CMP_Project_Info`) – Instance of *CMP\_Project\_Info* object
- **pipeline** (`cmp.pipelines.functional.fMRI.fMRIPipeline`) – Instance of *fMRIPipeline* object

## 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**

**check\_config()**

Old method that was checking custom settings (obsolete).

**clear\_stages\_outputs()**

Clear processing stage outputs.

**create\_stage\_flow** (*stage\_name*)

Create the sub-workflow of a processing stage.

**Parameters** *stage\_name* –

**fill\_stages\_outputs()**

Update processing stage output list for visual inspection.

**launch\_process()**

Launch the processing.

**number\_of\_cores = 1**

**subject = 'sub-01'**

**class** cmp.pipelines.common.**ProcessThread** (*group=None, target=None, name=None, args=(), kwargs=None, \*, daemon=None*)

Bases: threading.Thread

Class use to represent the pipeline process as a threading.Thread.

**pipeline** <Instance>

Any Pipeline instance

**pipeline** = <traits.trait\_types.Instance object>

**run()**

Execute the pipeline.

**class** cmp.pipelines.common.**ProgressThread** (*group=None, target=None, name=None, args=(), kwargs=None, \*, daemon=None*)

Bases: threading.Thread

Class use to monitor stage execution in a threading.Thread.

Information is display in the ProgressWindow (Not used anymore by CMP3)

**pw** = <traits.trait\_types.Instance object>

**run()**  
Monitors stage execution a workflow.

**stage\_names** = []

**stages** = {}

**class** cmp.pipelines.common.**ProgressWindow**  
Bases: traits.has\_traits.HasTraits  
Progress window of stage execution  
(Not used anymore by CMP3)

## 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 input 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\_pipeline\_flow** (*cmp\_deriv\_subject\_directory*, *nipype\_deriv\_subject\_directory*)  
Create the pipeline workflow.

**Parameters**

- **<Directory>** (*nipype\_deriv\_subject\_directory*) – Main CMP output directory of a subject e.g. /output\_dir/cmp/sub-XX/ (ses-YY)
- **<Directory>** – Intermediate Nipype output directory of a subject e.g. /output\_dir/nipype/sub-XX/ (ses-YY)

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

**Return type** `anat_flow` <`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.Global_Configuration` object>

**input\_folders** = ['anat']

**now** = '20210115\_1538'

**ordered\_stage\_list** = ['Segmentation', 'Parcellation']

**process** ()

Executes the pipeline workflow and returns True if successful.

**update\_parcellation** ()

Update self.stages['Parcellation'].config.parcellation\_scheme when seg\_tool is updated.

**update\_parcellation\_scheme** ()

Updates parcellation\_scheme and atlas\_info when parcellation\_scheme is updated.

**update\_segmentation** ()

Update self.stages['Segmentation'].config.seg\_tool when parcellation\_scheme is updated.

**class** `cmp.pipelines.anatomical.anatomical.Check_Input_Notification`  
Bases: `traits.has_traits.HasTraits`

**class** `cmp.pipelines.anatomical.anatomical.Global_Configuration`  
Bases: `traits.has_traits.HasTraits`

Global pipeline configurations.

**process\_type**

Processing pipeline type

**Type** 'fMRI'

**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

Diffusion pipeline Class definition.

**class** cmp.pipelines.diffusion.diffusion.Check\_Input\_Notification

Bases: traits.has\_traits.HasTraits

**class** cmp.pipelines.diffusion.diffusion.DiffusionPipeline (*project\_info*)

Bases: *cmp.pipelines.common.Pipeline*

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

It is composed of the preprocessing stage that preprocesses dMRI, the registration stage that co-registers T1w to the diffusion B0 and projects the parcellations to the native diffusion space, the diffusion stage that estimates tensors or fiber orientation distributions functions from the diffusion signal and reconstructs fiber using tractography, and finally the connectome stage that combines the output tractogram with the parcellations to create the structural connectivity matrices.

**See also:**

*cmp.stages.preprocessing.preprocessing.PreprocessingStage, cmp.stages.registration.registration.RegistrationStage, cmp.stages.diffusion.diffusion.DiffusionStage, cmp.stages.connectome.connectome.ConnectomeStage*

**check\_config()**

Check if the list output formats in the configuration of the connectome stage is not empty.

**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\_pipeline\_flow** (*cmp\_deriv\_subject\_directory, nipy\_deriv\_subject\_directory*)

Create the pipeline workflow.

**Parameters**

- **<Directory>** (*nipy\_deriv\_subject\_directory*) – Main CMP output directory of a subject e.g. /output\_dir/cmp/sub-XX/ (ses-YY)



- **<Directory>** – Intermediate Nipype output directory of a subject e.g. `output_dir/nipype/sub-XX/ (ses-YY)`

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

**Return type** `diffusion_flow <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”, “Diffusion” and “Connectome”.

`global_conf = <cmp.pipelines.diffusion.diffusion.Global_Configuration object>`

`input_folders = ['anat', 'dwi']`

`now = '20210115_1538'`

`ordered_stage_list = ['Preprocessing', 'Registration', 'Diffusion', 'Connectome']`

`process()`

Executes the pipeline workflow and returns True if successful.

`update_outputs_recon` (*new*)

Update list of of outputs of the diffusion stage when `recon_processing_tool` is updated.

**Parameters** `new` – New value.

`update_outputs_tracking` (*new*)

Update list of of outputs of the diffusion stage when `tracking_processing_tool` is updated.

**Parameters** `new` – New value.

`update_vizualization_layout` (*new*)

Update list of of outputs of the connectome stage when `circular_layout` is updated.

**Parameters** `new` – New value.

`update_vizualization_logscale` (*new*)

Update list of of outputs of the connectome stage when `log_visualization` is updated.

**Parameters** `new` – New value.

**class** `cmp.pipelines.diffusion.diffusion.Global_Configuration`

Bases: `traits.has_traits.HasTraits`

Global pipeline configurations.

**process\_type**

Processing pipeline type

**Type** ‘fMRI’

**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

**modalities**

List of available diffusion modalities read from the `acq-<modality>` filename keyword

**Type** traits.List

**dmri\_bids\_acq**

Diffusion modality to be processed

**Type** traits.Str

**cmp.pipelines.functional package****Submodules****cmp.pipelines.functional.eeg module**

EEG pipeline Class definition

**cmp.pipelines.functional.fMRI module**

Functional pipeline Class definition.

**class** `cmp.pipelines.functional.fMRI.Check_Input_Notification`

Bases: `traits.has_traits.HasTraits`

**class** `cmp.pipelines.functional.fMRI.Global_Configuration`

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, deskipping 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 (`FunctionalMRISage`) 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\_input** (*layout*, *gui=True*, *debug=False*)

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\_pipeline\_flow** (*cmp\_deriv\_subject\_directory*, *nipype\_deriv\_subject\_directory*)

Create the pipeline workflow.

**Parameters**

- **<Directory>** (*nipype\_deriv\_subject\_directory*) – Main CMP output directory of a subject e.g. `/output_dir/cmp/sub-XX/ (ses-YY)`
- **<Directory>** – Intermediate Nipype output directory of a subject e.g. `/output_dir/nipype/sub-XX/ (ses-YY)`

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

**Return type** `fMRI_flow <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.Global_Configuration object>`

**input\_folders** = `['anat', 'func']`

**now** = `'20210115_1538'`

**ordered\_stage\_list** = `['Preprocessing', 'Registration', 'FunctionalMRI', 'Connectome']`

**process** ()

**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', 'cff', '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.

**has\_run** ()

Function that returns True if the stage has been run successfully.

#### Returns

**Return type** True if the stage has been run successfully

### `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 workflow.

**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.

**has\_run()**

Function that returns `True` if the stage has been run successfully.

**Returns**

**Return type** `True` if the stage has been run successfully

## cmp.stages.diffusion package

### Submodules

#### cmp.stages.diffusion.diffusion module

Definition of config and stage classes for diffusion reconstruction and tractography.

**class** cmp.stages.diffusion.diffusion.**DiffusionConfig**

Bases: traits.has\_traits.HasTraits

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

**diffusion\_imaging\_model\_editor**

Available diffusion imaging models

**Type** ['DSI', 'DTI', 'HARDI']

**diffusion\_imaging\_model**

Selected diffusion imaging model (Default: 'DTI')

**Type** traits.Str

**dilate\_rois**

Dilate parcellation regions-of-interest (Default: True)

**Type** traits.Bool

**dilation\_kernel**

Type of dilation kernel to used

**Type** traits.Enum(['Box', 'Gauss', 'Sphere'])

**dilation\_radius**

Radius of the dilation kernel

**Type** traits.Enum([1, 2, 3, 4])

**recon\_processing\_tool\_editor**

List of processing tools available for diffusion signal reconstruction

**Type** ['Dipy', 'MRtrix']

**tracking\_processing\_tool\_editor**

List of processing tools available for tractography

**Type** ['Dipy', 'MRtrix']

**processing\_tool\_editor**

List of processing tools available for diffusion signal reconstruction and tractography

**Type** ['Dipy', 'MRtrix']

**recon\_processing\_tool**

Processing tool to use for diffusion signal modeling (Default: 'MRtrix')

**Type** traits.Str

**tracking\_processing\_tool**

Processing tool to use for tractography (Default: 'MRtrix')

**Type** traits.Str

**custom\_track\_file**

Custom tractogram file to used as input to the connectome stage (obsolete)

**Type** traits.File

**dipy\_recon\_config**

Configuration instance of the Dipy reconstruction stage

**Type** Instance(HasTraits)

**mrtrix\_recon\_config**

Configuration instance of the MRtrix3 reconstruction stage

**Type** Instance(HasTraits)

**dipy\_tracking\_config**

Configuration instance of the Dipy tracking (tractography) stage

**Type** Instance(HasTraits)

**mrtrix\_tracking\_config**

Configuration instance of the MRtrix3 tracking (tractography) stage

**Type** Instance(HasTraits)

**diffusion\_model\_editor**

List of types of available local tractography algorithms.

**Type** ['Deterministic', 'Probabilistic']

**diffusion\_model**

Type of local tractography algorithm to use. (Default: 'Probabilistic')

**Type** traits.Str

**See also:**

`cmp.stages.diffusion.reconstruction.Dipy_recon_config`, `cmp.stages.diffusion.reconstruction.MRtrix_recon_config`, `cmp.stages.diffusion.tracking.Dipy_tracking_config`, `cmp.stages.diffusion.tracking.MRtrix_tracking_config`, `cmp.stages.diffusion.diffusion.DiffusionStage`

**update\_dipy\_tracking\_SD** (*new*)

Update SD of `dipy_tracking_config` when `local_model` is updated.

**Parameters** *new* (*string*) – New value of `local_model`

**update\_dipy\_tracking\_sh\_order** (*new*)

Update `sh_order` of `dipy_tracking_config` when `lmax_order` is updated.

**Parameters** *new* (*int*) – New value of `lmax_order`

**update\_mrtrix\_tracking\_SD** (*new*)

Update SD of `mrtrix_tracking_config` when `local_model` is updated.

**Parameters** *new* (*string*) – New value of `local_model`

**class** `cmp.stages.diffusion.diffusion.DiffusionStage` (*bids\_dir*, *output\_dir*)

Bases: `cmp.stages.common.Stage`

Class that represents the diffusion stage of a `DiffusionPipeline`.

The diffusion stage workflow is composed of two sub-workflows: 1. `recon_flow` that estimates tensors or fiber orientation distribution functions from dMRI, 2. `track_flow` that runs tractography from the output of `recon_flow`.



**create\_workflow()**

Create the workflow of the *DiffusionStage*

See also:

```
cmp.pipelines.diffusion.diffusion.DiffusionPipeline,          cmp.stages.
diffusion.diffusion.DiffusionConfig,    cmp.stages.diffusion.reconstruction.
Dipy_recon_config, cmp.stages.diffusion.reconstruction.MRtrix_recon_config,
cmp.stages.diffusion.tracking.Dipy_tracking_config,          cmp.stages.
diffusion.tracking.MRtrix_tracking_config,          cmp.stages.diffusion.
reconstruction.create_dipy_recon_flow,          cmp.stages.diffusion.
reconstruction.create_mrtrix_recon_flow,          cmp.stages.diffusion.
tracking.create_dipy_tracking_flow,          cmp.stages.diffusion.tracking.
create_mrtrix_tracking_flow
```

**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

See also:

```
cmp.stages.diffusion.reconstruction.create_dipy_recon_flow(),    cmp.
stages.diffusion.reconstruction.create_mrtrix_recon_flow(), cmp.stages.
diffusion.tracking.create_dipy_tracking_flow(),    cmp.stages.diffusion.
tracking.create_mrtrix_tracking_flow()
```

**define\_inspect\_outputs()**

Update the 'inspect\_outputs' class attribute.

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

**has\_run()**

Function that returns True if the stage has been run successfully.

**Returns**

**Return type** True if the stage has been run successfully

```
cmp.stages.diffusion.diffusion.strip_suffix(file_input, prefix)
```

Extract path of file\_input and add prefix to generate a prefix path for outputs.

**Parameters**

- **file\_input** (*os.path.abspath*) – Absolute path to an input file
- **prefix** (*os.path*) – Prefix to used in the generation of the output prefix path.

**Returns** out\_prefix\_path – The generated prefix path

**Return type** os.path

**cmp.stages.diffusion.reconstruction module**

Reconstruction methods and workflows.

```
class cmp.stages.diffusion.reconstruction.Dipy_recon_config
    Bases: traits.has_traits.HasTraits

    Class used to store Dipy diffusion reconstruction sub-workflow configuration parameters.

    imaging_model
        Diffusion imaging model (For instance 'DTI')
        Type Str

    flip_table_axis
        Axis to be flipped in the gradient table.
        Type traits.List(['x', 'y', 'z'])

    local_model_editor
        List of reconstruction models
        Type {False: '1:Tensor', True: '2:Constrained Spherical Deconvolution'}

    local_model
        Reconstruction model selected (See local_model_editor) (Default: True, meaning Tensor is performed)
        Type traits.Bool

    lmax_order
        Choices of maximal order to use for Constrained Spherical Deconvolution
        Type traits.Enum([2, 4, 6, 8, 10, 12, 14, 16])

    single_fib_thr
        FA threshold
        Type traits.Float(0.7, min=0, max=1)

    recon_mode
        Can be "Probabilistic" or "Deterministic"
        Type traits.Str

    mapmri
        Type traits.Bool(False)

    tracking_processing_tool
        Type traits.Enum('MRtrix', 'Dipy')

    laplacian_regularization
        Apply laplacian regularization in MAP-MRI if True (Default: True)
        Type traits.Bool

    laplacian_weighting
        Laplacian regularization weight in MAP-MRI (Default: 0.05)
        Type traits.Float

    positivity_constraint
        Apply positivity constraint in MAP-MRI if True (Default: True)
        Type traits.Bool
```

**radial\_order**

MAP-MRI radial order (Default: 8)

**Type** traits.Int

**small\_delta**

Small data for gradient table (pulse duration) used by MAP-MRI (Default: 0.02)

**Type** traits.Float

**big\_delta**

Big data for gradient table (time interval) used by MAP-MRI (Default: 0.5)

**Type** traits.Float

**radial\_order\_values**

Choices of radial order values used by SHORE

**Type** traits.List([2, 4, 6, 8, 10, 12])

**shore\_radial\_order**

Even number that represents the order of the basis (Default: 6)

**Type** traits.Str

**shore\_zeta**

Scale factor in SHORE (Default: 700)

**Type** traits.Int

**shore\_lambda\_n**

Radial regularisation constant in SHORE (Default: 1e-8)

**Type** traits.Float

**shore\_lambda\_l**

Angular regularisation constant in SHORE (Default: 1e-8)

**Type** traits.Float

**shore\_tau**

Diffusion time used by SHORE. By default the value that makes  $q$  equal to the square root of the b-value (Default: 0.025330295910584444)

**Type** traits.Float

**shore\_constrain\_e0**

Constrain SHORE optimization such that  $E(0) = 1$  (Default: False)

**Type** traits.Bool

**shore\_positive\_constraint**

Constrain the SHORE propagator to be positive (Default: False)

**Type** traits.Bool

**class** cmp.stages.diffusion.reconstruction.MRtrix\_recon\_config

Bases: traits.has\_traits.HasTraits

Class used to store Dipy diffusion reconstruction sub-workflow configuration parameters.

**flip\_table\_axis**

Axis to be flipped in the gradient table.

**Type** traits.List(['x', 'y', 'z'])

**local\_model\_editor**

List of reconstruction models

**Type** {False: '1:Tensor', True: '2:Constrained Spherical Deconvolution'}

**local\_model**

Reconstruction model selected (See [local\\_model\\_editor](#)) (Default: True, meaning Tensor is performed)

**Type** traits.Bool

**lmax\_order**

Choices of maximal order to use for Constrained Spherical Deconvolution

**Type** traits.Enum([2, 4, 6, 8, 10, 12, 14, 16])

**single\_fib\_thr**

FA threshold

**Type** traits.Float(0.7, min=0, max=1)

**recon\_mode**

Can be “Probabilistic” or “Deterministic”

**Type** traits.Str

`cmp.stages.diffusion.reconstruction.create_dipy_recon_flow(config)`

Create the reconstruction sub-workflow of the DiffusionStage using Dipy.

**Parameters** `config` ([Dipy\\_recon\\_config](#)) – Workflow configuration

**Returns** `flow` – Built reconstruction sub-workflow

**Return type** `nipype.pipeline.engine.Workflow`

`cmp.stages.diffusion.reconstruction.create_mrtrix_recon_flow(config)`

Create the reconstruction sub-workflow of the DiffusionStage using MRtrix3.

**Parameters** `config` ([Dipy\\_recon\\_config](#)) – Workflow configuration

**Returns** `flow` – Built reconstruction sub-workflow

**Return type** `nipype.pipeline.engine.Workflow`

## **cmp.stages.diffusion.tracking module**

Tracking methods and workflows of the diffusion stage.

**class** `cmp.stages.diffusion.tracking.Dipy_tracking_config`

Bases: `traits.has_traits.HasTraits`

Class used to store Dipy diffusion reconstruction sub-workflow configuration parameters.

**imaging\_model**

Diffusion imaging model (For example ‘DTI’)

**Type** traits.Str

**tracking\_mode**

Type of local tractography algorithm (Can be “Deterministic” or “Probabilistic”)

**Type** traits.Str

**SD**

If True, inputs are coming from Constrained Spherical Deconvolution reconstruction

**Type** traits.Bool

**number\_of\_seeds**

Number of seeds (Default: 1000)

**Type** traits.Int

**seed\_density**

Number of seeds to place along each direction where a density of 2 is the same as [2, 2, 2] and will result in a total of 8 seeds per voxel (Default: 1.0)

**Type** traits.Float

**fa\_thresh**

Fractional Anisotropy (FA) threshold (Default: 0.2)

**Type** traits.Float

**step\_size**

Tractography algorithm step size (Default: 0.5)

**Type** traits.trait.Float

**max\_angle**

Maximum streamline angle allowed (Default: 25.0)

**Type** traits.Float

**sh\_order**

Order used for Constrained Spherical Deconvolution reconstruction (Default: 8)

**Type** traits.Int

**use\_act**

Use FAST for partial volume estimation and Anatomically-Constrained Tractography (ACT) tissue classifier (Default: False)

**Type** traits.Bool

**seed\_from\_gmwm**

Seed from Grey Matter / White Matter interface (requires Anatomically-Constrained Tractography (ACT)) (Default: False)

**Type** traits.Bool

**class** cmp.stages.diffusion.tracking.MRtrix\_tracking\_config

Bases: traits.has\_traits.HasTraits

Class used to store Dipy diffusion reconstruction sub-workflow configuration parameters.

**tracking\_mode**

Type of local tractography algorithm (Can be “Deterministic” or “Probabilistic”)

**Type** traits.Str

**SD**

If True, inputs are coming from Constrained Spherical Deconvolution reconstruction

**Type** traits.Bool

**desired\_number\_of\_tracks**

Desired number of output streamlines in the tractogram (Default: 1M)

**Type** traits.Int

**curvature = Float**

Maximum streamline curvature (Default: 2.0)

**min\_length = Float**

Minimal streamline length (Default: 5)

**max\_length = Float**

Maximal streamline length (Default: 500)

**angle**

Maximum streamline angle allowed (Default: 45.0)

**Type** traits.Float

**cutoff\_value**

Cut-off value to terminate streamline (Default: 0.05)

**Type** traits.Float

**use\_act**

Use `5ttgen` for brain tissue types estimation and Anatomically-Constrained Tractography (ACT) tissue classifier (Default: False)

**Type** traits.Bool

**seed\_from\_gmwmi**

Seed from Grey Matter / White Matter interface (requires Anatomically-Constrained Tractography (ACT)) (Default: False)

**Type** traits.Bool

**crop\_at\_gmwmi**

Crop streamline endpoints more precisely as they cross the GM-WM interface (requires Anatomically-Constrained Tractography (ACT)) (Default: True)

**Type** traits.Bool

**backtrack**

Allow tracks to be truncated (requires Anatomically-Constrained Tractography (ACT)) (Default: True)

**Type** traits.Bool

**sift**

Filter tractogram using mrtrix3 SIFT (Default: True)

**Type** traits.Bool

`cmp.stages.diffusion.tracking.create_dipy_tracking_flow(config)`

Create the tractography sub-workflow of the `DiffusionStage` using Dipy.

**Parameters** `config` (`Dipy_tracking_config`) – Sub-workflow configuration object

**Returns** `flow` – Built tractography sub-workflow

**Return type** `nipype.pipeline.engine.Workflow`

`cmp.stages.diffusion.tracking.create_mrtrix_tracking_flow(config)`

Create the tractography sub-workflow of the `DiffusionStage` using MRtrix3.

**Parameters** `config` (`MRtrix_tracking_config`) – Sub-workflow configuration object

**Returns** `flow` – Built tractography sub-workflow

**Return type** `nipype.pipeline.engine.Workflow`

`cmp.stages.diffusion.tracking.get_freesurfer_parcellation(roi_files)`  
 Return the first file in the list of parcellation files

**Parameters** `roi_files` (*list of traits.File*) – List of parcellation files

## 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")

**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, out-put_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 workflow.

#### 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.

**has\_run()**

Function that returns True if the stage has been run successfully.

#### Returns

**Return type** True if the stage has been run successfully

## 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: 'Lausanne2008')

**Type** traits.Str

**parcellation\_scheme\_editor**

Choice of parcellation schemes

**Type** traits.List(['NativeFreesurfer', 'Lausanne2008', 'Lausanne2018'])



**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': number\_of\_regions, >>> 'node\_information\_graphml': graphml\_file}} # doctest: +SKIP

**Type** traits.Dict

**See also:**

`cmp.stages.parcellation.parcellation.ParcellationStage`

**update\_atlas\_info()**

Update `atlas_info` class attribute.

```
class cmp.stages.parcellation.parcellation.ParcellationStage(pipeline_mode,
                                                             bids_dir,      out-
                                                             put_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 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.

**has\_run()**

Function that returns `True` if the stage has been run successfully.

**Returns**

**Return type** `True` if the stage has been run successfully

## 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*,  
*out\_dir*,  
*put\_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 workflow.

#### 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.

**has\_run()**

Function that returns True if the stage has been run successfully.

#### Returns

**Return type** True if the stage has been run successfully

## cmp.stages.preprocessing.preprocessing module

CMP preprocessing Stage (not used yet!)

**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

**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 workflow.

#### Parameters

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

**has\_run** ()

Function that returns `True` if the stage has been run successfully.

#### Returns

**Return type** `True` if the stage has been run successfully

## 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: 'spm')

**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

**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="", out-  
                                                             put_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\_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.



**has\_run()**

Function that returns `True` if the stage has been run successfully.

**Returns**

**Return type** `True` if the stage has been run successfully

`cmp.stages.registration.registration.unicode2str(text)`

Convert a unicode to a string using system's encoding.

(This methods should be moved to utils in the upcoming future.)

**Parameters** `text` (*bytes*) – Unicode bytes representation of a string

**Returns** `out_str` – Output string

**Return type** `str`

## 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"])`

**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

**brain\_mask\_path**

Custom brain mask path

**Type** traits.File

**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\_trait**

Freesurfer subject (being processed) directory path usually /output\_dir/freesurfer/sub-XX(\_ses-YY)

**Type** traits.List

**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

**white\_matter\_mask**

Custom white-matter mask

**Type** traits.File

**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*

**class** cmp.stages.segmentation.segmentation.**SegmentationStage** (*bids\_dir*, *out-put\_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 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.

**has\_run()**

Function that returns True if the stage has been run successfully.

#### Returns

**Return type** True if the stage has been run successfully

*cmp.stages.segmentation.segmentation.extract\_base\_directory*(*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

## 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\_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.FunctionalMRISage, 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.core module

Core module providing a `run()` command.

`cmp.bidsappmanager.core.run(command, env=None, cwd=None)`

Function calls by `CMP_MainWindowHandler` to run datalad commands.

It runs the command specified as input via `subprocess.run()`.

**Parameters**

- **command** (*string*) – String containing the command to be executed (required)
- **env** (*os.environ*) – Specify a custom `os.environ`
- **cwd** (*os.path*) – Specify a custom current working directory

## Examples

```
>>> cmd = 'datalad save'
>>> run(cmd)
```

## cmp.bidsappmanager.gui module

Connectome Mapper GUI.

```
class cmp.bidsappmanager.gui.CMP_BIDSAppWindow(project_info=None,
                                                bids_root="",          subjects=None,
                                                list_of_subjects_to_be_processed=None,
                                                anat_config="",          dmri_config="",
                                                fmri_config="")
```

Bases: `traits.has_traits.HasTraits`

Class that defines the Window of the BIDS App Interface.

### **project\_info**

Instance of `CMP_Project_Info` that represents the processing project

**Type** `CMP_Project_Info`

### **bids\_root**

BIDS root dataset directory

**Type** `traits.Directory`

### **output\_dir**

Output directory

**Type** `traits.Directory`

### **subjects**

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

**Type** `traits.List`

### **number\_of\_participants\_processed\_in\_parallel**

Number of participants / subjects to be processed in parallel that takes values in the `[1, # of CPUs - 1]` range

**Type** `traits.Range`

### **number\_threads\_max**

Maximal number of threads to be used by OpenMP programs (4 by default)

**Type** `traits.Int`

### **number\_of\_threads**

Number of threads to be used by OpenMP programs that takes values in the `[1, number_threads_max]` range

**Type** `traits.Range`

### **fs\_file**

Path to Freesurfer license file

**Type** `traits.File`

### **list\_of\_subjects\_to\_be\_processed**

Selection of subjects to be processed from the `subjects` list

**Type** List(Str)

**dmri\_inputs\_checked**

True if dMRI data is available in the dataset

**Type** traits.Bool

**fmri\_inputs\_checked**

True if fMRI data is available in the dataset

**Type** traits.Bool

**anat\_config**

Configuration file for the anatomical MRI pipeline

**Type** traits.File

**dmri\_config**

Configuration file for the diffusion MRI pipeline

**Type** traits.File

**fmri\_config**

Configuration file for the functional MRI pipeline

**Type** traits.File

**run\_anat\_pipeline**

If True, run the anatomical pipeline

**Type** traits.Bool

**run\_dmri\_pipeline**

If True, run the diffusion pipeline

**Type** traits.Bool

**run\_fmri\_pipeline**

If True, run the functional pipeline

**Type** traits.Bool

**bidsapp\_tag**

Selection of BIDS App version to use

**Type** traits.Enum

**data\_provenance\_tracking**

If set and if datalad\_is\_available is True run the BIDS App using datalad (False by default)

**Type** traits.Bool

**datalad\_update\_environment**

If True and data\_provenance\_tracking is True, tell to datalad to update the BIDS App container image if there was a previous execution (True by default)

**Type** traits.Bool

**datalad\_is\_available**

Boolean used to store if datalad is available in the computing environment (False by default)

**Type** traits.Bool

**check**

Button to check if all parameters are properly set for execution of the BIDS App

**Type** traits.ui.Button

**start\_bidsapp**

Button to run the BIDS App

**Type** `traits.ui.Button`

**traits\_view**

TraitsUI QtView that describes the content of the window

**Type** `QtView`

**check\_settings()**

Checks if all the parameters of the BIDS App run are properly set before execution.

**classmethod manage\_bidsapp\_procs(*proclist*)**

Manage parallelized process at the participant level

**Parameters** *proclist* (*List of subprocess.Popen*) – List of Popen processes

**number\_of\_parallel\_procs\_updated(*new*)**

Callback function when `number_of_parallel_procs` is updated.

**classmethod run(*command*, *env=None*, *cwd='/home/docs/checkouts/readthedocs.org/user\_builds/connectome-mapper-3/checkouts/v3.0.0-rc2/docs'*)**

Function to run datalad commands.

It runs the command specified as input via `subprocess.run()`.

**Parameters**

- **command** (*string*) – String containing the command to be executed (required)
- **env** (*os.environ*) – Specify a custom `os.environ`
- **cwd** (*os.path*) – Specify a custom current working directory

**Examples**

```
>>> cmd = 'datalad save -m my dataset change message'
>>> run(cmd)
```

**start\_bids\_app()**

Function executed when the Run BIDS App button is clicked.

It implements all steps in the creation and execution of the BIDS App with or without datalad.

**start\_bidsapp\_participant\_level\_process(*bidsapp\_tag*, *participant\_labels*)**

Create and run the BIDS App command.

**Parameters**

- **bidsapp\_tag** (*traits.Str*) – Version tag of the CMP 3 BIDS App
- **participant\_labels** (*traits.List*) – List of participants labels in the form ["01", "03", "04", ...]

**start\_bidsapp\_participant\_level\_process\_with\_datalad(*bidsapp\_tag*, *participant\_labels*)**

Create and run the BIDS App command with Datalad.

**Parameters**

- **bidsapp\_tag** (*traits.Str*) – Version tag of the CMP 3 BIDS App

- **participant\_labels** (*traits.List*) – List of participants labels in the form ["01", "03", "04", ...]

**update\_checksettings** (*new*)

Function that reset `settings_checked` attribute to False.

**update\_run\_anat\_pipeline** (*new*)

Callback function when `run_anat_pipeline` is updated.

**update\_run\_dmri\_pipeline** (*new*)

Callback function when `run_dmri_pipeline` is updated.

**update\_run\_fmri\_pipeline** (*new*)

Callback function when `run_fmri_pipeline` is updated.

```
class cmp.bidsappmanager.gui.CMP_ConfiguratorWindow(project_info=None,  
                                                    anat_pipeline=None,  
                                                    dmri_pipeline=None,  
                                                    fmri_pipeline=None,  
                                                    anat_inputs_checked=False,  
                                                    dmri_inputs_checked=False,  
                                                    fmri_inputs_checked=False)
```

Bases: `traits.has_traits.HasTraits`

Class that defines the Configurator Window.

**project\_info**

Instance of `CMP_Project_Info` that represents the processing project

**Type** *CMP\_Project\_Info*

**anat\_pipeline**

Instance of anatomical MRI pipeline UI

**Type** `Instance(HasTraits)`

**dmri\_pipeline**

Instance of diffusion MRI pipeline UI

**Type** `Instance(HasTraits)`

**fmri\_pipeline**

Instance of functional MRI pipeline UI

**Type** `Instance(HasTraits)`

**anat\_inputs\_checked**

Boolean that indicates if anatomical pipeline inputs are available (Default: False)

**dmri\_inputs\_checked** = `[traits.Bool]` Boolean that indicates if diffusion pipeline inputs are available (Default: False)

**fmri\_inputs\_checked** `[traits.Bool]` Boolean that indicates if functional pipeline inputs are available (Default: False)

**Type** `traits.Bool`

**anat\_save\_config**

TraitsUI Action to save the anatomical pipeline configuration

**Type** `traits.ui.Action`



**dmri\_save\_config**

TraitsUI Action to save the diffusion pipeline configuration

**Type** traits.ui.Action

**fmri\_save\_config**

TraitsUI Action to save the functional pipeline configuration

**Type** traits.ui.Action

**save\_all\_config**

Button to save all configuration files at once

**Type** traits.ui.Button

**traits\_view**

TraitsUI QtView that describes the content of the window

**Type** QtView

**update\_diffusion\_imaging\_model** (*new*)

```
class cmp.bidsappmanager.gui.CMP_InspectorWindow (project_info=None,
                                                    anat_inputs_checked=False,
                                                    dmri_inputs_checked=False,
                                                    fmri_inputs_checked=False)
```

Bases: traits.has\_traits.HasTraits

Class that defines the Configurator Window.

**project\_info**

Instance of CMP\_Project\_Info that represents the processing project

**Type** *CMP\_Project\_Info*

**anat\_pipeline**

Instance of anatomical MRI pipeline

**Type** Instance(HasTraits)

**dmri\_pipeline**

Instance of diffusion MRI pipeline

**Type** Instance(HasTraits)

**fmri\_pipeline**

Instance of functional MRI pipeline

**Type** Instance(HasTraits)

**anat\_inputs\_checked**

Indicates if inputs of anatomical pipeline are available (Default: False)

**Type** traits.Bool

**dmri\_inputs\_checked**

Indicates if inputs of diffusion pipeline are available (Default: False)

**Type** traits.Bool

**fmri\_inputs\_checked**

Indicates if inputs of functional pipeline are available (Default: False)

**Type** traits.Bool

**output\_anat\_available**

Indicates if outputs of anatomical pipeline are available (Default: False)

**Type** traits.Bool

**output\_dmri\_available**

Indicates if outputs of diffusion pipeline are available (Default: False)

**Type** traits.Bool

**output\_fmri\_available**

Indicates if outputs of functional pipeline are available (Default: False)

**Type** traits.Bool

**traits\_view**

TraitsUI QtView that describes the content of the window

**Type** QtView

**select\_subject()**

Function to select the subject and session for which to inspect outputs.

**update\_diffusion\_imaging\_model** (*new*)

Function called when `diffusion_imaging_model` is updated.

**class** cmp.bidsappmanager.gui.CMP\_MainWindow

Bases: traits.has\_traits.HasTraits

Class that defines the Main window of the Connectome Mapper 3 GUI.

**project\_info**

Instance of *CMP\_Project\_InfoUI* that represents the processing project

**Type** *CMP\_Project\_InfoUI*

**anat\_pipeline**

Instance of anatomical MRI pipeline UI

**Type** Instance(HasTraits)

**dmri\_pipeline**

Instance of diffusion MRI pipeline UI

**Type** Instance(HasTraits)

**fmri\_pipeline**

Instance of functional MRI pipeline UI

**Type** Instance(HasTraits)

**bidsapp\_ui**

Instance of *CMP\_BIDSAppWindow*

**Type** *CMP\_Project\_Info*

**load\_dataset**

TraitsUI Action to load a BIDS dataset

**Type** traits.ui.Action

**bidsapp**

Button that displays the BIDS App Interface window

**Type** traits.ui.Button

**configurator**

Button that displays the pipeline Configurator window

**Type** traits.ui.Button

**quality\_control**

Button that displays the pipeline Quality Control / Inspector window

**Type** traits.ui.Button

**manager\_group**

TraitsUI View that describes the content of the main window

**Type** traits.ui.View

**traits\_view**

TraitsUI QtView that includes `manager_group` and parameterize the window with menu

**Type** QtView

**show\_bidsapp\_interface()**

Callback of the “BIDS App” button. This displays the BIDS App Interface Window.

**view\_mode** = 1

**class** cmp.bidsappmanager.gui.CMP\_Project\_InfoUI

Bases: `cmp.project.CMP_Project_Info`

Class that extends the `CMP_Project_Info` with graphical components.

It supports graphically the setting of all processing properties / attributes of an `CMP_Project_Info` instance.

**creation\_mode**

Mode for loading the dataset. Valid values are ‘Load BIDS dataset’, ‘Install Datalad BIDS dataset’

**Type** traits.Enum

**install\_datalad\_dataset\_via\_ssh**

If set to True install the datalad dataset from a remote server via ssh.(True by default)

**Type** traits.Bool

**ssh\_user**

Remote server username. (Required if `install_datalad_dataset_via_ssh` is True)

**Type** traits.Str

**ssh\_pwd <traits.Password>**

Remote server password. (Required if `install_datalad_dataset_via_ssh` is True)

**ssh\_remote**

Remote server IP or URL. (Required if `install_datalad_dataset_via_ssh` is True)

**Type** traits.Str

**datalad\_dataset\_path**

Path to the datalad dataset on the remote server. (Required if `install_datalad_dataset_via_ssh` is True)

**Type** traits.Directory

**summary\_view\_button**

Button that shows the pipeline processing summary table

**Type** traits.ui.Button

**pipeline\_processing\_summary\_view**

TraitsUI VGroup that contains `Item('pipeline_processing_summary')`

**Type** traits.ui.VGroup

**dataset\_view**

TraitsUI View that shows a summary of project settings and modality available for a given subject

**Type** traits.ui.View

**traits\_view**

TraitsUI QtView that includes the View 'dataset\_view'

**Type** QtView

**create\_view**

Dialog view to create a BIDS Dataset

**Type** traits.ui.View

**subject\_view**

Dialog view to select of subject

**Type** traits.ui.View

**subject\_session\_view**

Dialog view to select the subject session

**Type** traits.ui.View

**dmri\_bids\_acq\_view**

Dialog view to select the diffusion acquisition model

**Type** traits.ui.View

**anat\_warning\_view**

View that displays a warning message regarding the anatomical T1w data

**Type** traits.ui.View

**anat\_config\_error\_view**

Error view that displays an error message regarding the configuration of the anatomical pipeline

**Type** traits.ui.View

**dmri\_warning\_view**

View that displays a warning message regarding the diffusion MRI data

**Type** traits.ui.View

**dmri\_config\_error\_view**

View that displays an error message regarding the configuration of the diffusion pipeline

**Type** traits.ui.View

**fmri\_warning\_view**

View that displays a warning message regarding the functional MRI data

**Type** traits.ui.View

**fmri\_config\_error\_view**

View that displays an error message regarding the configuration of the fMRI pipeline

**Type** traits.ui.View

**open\_view**

Dialog view to load a BIDS Dataset

**Type** traits.ui.View

**anat\_select\_config\_to\_load**

Dialog view to load the configuration file of the anatomical pipeline

**Type** traits.ui.View

**diffusion\_imaging\_model\_select\_view**

Dialog view to select the diffusion acquisition model

**Type** traits.ui.View

**dmri\_select\_config\_to\_load**

Dialog view to load the configuration file of the diffusion MRI pipeline

**Type** traits.ui.View

**fmri\_select\_config\_to\_load**

Dialog view to load the configuration file of the fMRI pipeline

**Type** traits.ui.View

**class** cmp.bidsappmanager.gui.MultiSelectAdapter

Bases: traitsui.tabular\_adapter.TabularAdapter

This adapter is used by both the left and right tables.

cmp.bidsappmanager.gui.get\_icon(*path*)

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

**Parameters** *path* (*string*) – Path to an image file

**Returns** *icon* – Return an instance of ImageResource or None if there is not graphical backend.

**Return type** ImageResource

## cmp.bidsappmanager.project module

Connectome Mapper Controller for handling GUI and non GUI general events.

**class** cmp.bidsappmanager.project.CMP\_BIDSAppWindowHandler

Bases: traitsui.handler.Handler

Event handler of the BIDS App Interface window.

**docker\_process**

Instance of subprocess.Popen where BIDS App docker image is run

**Type** subprocess.Popen

**check\_settings** (*ui\_info*)

Function that checks if all parameters are properly set before execution of the BIDS App.

**Parameters** *ui\_info* (*QtView*) – TraitsUI QtView associated with self

**classmethod** manage\_bidsapp\_procs (*proclist*)

Function that managed the parallelized BIDS App Popen process.

**Parameters** *proclist* – List of Popen processes running the BIDS App on a single subject

**start\_bids\_app** (*ui\_info*)

Main function that runs the BIDS App on a set or sub-set of participants.

**Parameters** *ui\_info* (*QtView*) – TraitsUI QtView associated with this handler

**classmethod** start\_bidsapp\_process (*participant\_label*)

Function that runs the BIDS App on a single subject.

**Parameters**

- **ui\_info** (*QtView*) – TraitsUI QtView associated with this handler
- **participant\_label** (*string*) – Label of the participant / subject (e.g. "01", no "sub-" prefix)

**classmethod stop\_bids\_app()**

Function that stops the BIDS execution.

**Parameters** **ui\_info** (*QtView*) – TraitsUI QtView associated with this handler

**class** cmp.bidsappmanager.project.**CMP\_ConfigQualityWindowHandler**

Bases: traitsui.handler.Handler

Event handler of the Configurator and Inspector (Quality Control) windows.

**project\_loaded**

Indicate if project has been successfully loaded (Default: False)

**Type** traits.Bool

**anat\_pipeline**

Instance of AnatomicalPipelineUI class

**Type** Instance(HasTraits)

**anat\_inputs\_checked**

Indicate if anatomical pipeline inputs are available (Default: False)

**Type** traits.Bool

**anat\_outputs\_checked**

Indicate if anatomical pipeline outputs are available (Default: False)

**Type** traits.Bool

**anatomical\_processed**

Indicate if anatomical pipeline was run (Default: False)

**Type** traits.Bool

**dmri\_pipeline**

Instance of DiffusionPipelineUI class

**Type** Instance(HasTraits)

**dmri\_inputs\_checked**

Indicate if diffusion pipeline inputs are available (Default: False)

**Type** traits.Bool

**dmri\_processed**

Indicate if diffusion pipeline was run (Default: False)

**Type** traits.Bool

**fmri\_pipeline**

Instance of fMRIPipelineUI class

**Type** Instance(HasTraits)

**fmri\_inputs\_checked**

Indicate if fMRI pipeline inputs are available (Default: False)

**Type** traits.Bool

**fmri\_processed**

Indicate if fMRI pipeline was run (Default: False)

Type traits.Bool

**load\_anat\_config\_file** (*ui\_info*)

Function that loads the anatomical pipeline configuration file.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with self

**load\_dmri\_config\_file** (*ui\_info*)

Function that loads the diffusion pipeline configuration file.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with self

**load\_fmri\_config\_file** (*ui\_info*)

Function that loads the fMRI pipeline configuration file.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with self

**load\_project** (*ui\_info*)

Function that creates a new CMP\_Project\_InfoUI instance from an existing project.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with self

**new\_project** (*ui\_info*)

Function that creates a new CMP\_Project\_InfoUI instance.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with self

**classmethod save\_anat\_config\_file** (*ui\_info*)

Function that saves the anatomical pipeline configuration file.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with self

**classmethod save\_dmri\_config\_file** (*ui\_info*)

Function that saves the diffusion pipeline configuration file.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with self

**classmethod save\_fmri\_config\_file** (*ui\_info*)

Function that saves the fMRI pipeline configuration file.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with self

**classmethod show\_bidsapp\_window** ()

Function that shows the BIDS App Interface Window.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with this handler

**update\_subject\_anat\_pipeline** (*ui\_info*)

Function that updates attributes of the AnatomicalPipelineUI instance.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with self

**update\_subject\_dmri\_pipeline** (*ui\_info*)

Function that updates attributes of the DiffusionPipelineUI instance.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with self

**update\_subject\_fmri\_pipeline** (*ui\_info*)

Function that updates attributes of the fMRIPipelineUI instance.

Parameters **ui\_info** (*QtView*) – TraitsUI QtView associated with self

**class** cmp.bidsappmanager.project.CMP\_MainWindowHandler

Bases: traitsui.handler.Handler

Event handler of the Configurator and Inspector (Quality Control) windows.

**project\_loaded**

Indicate if project has been successfully loaded (Default: False)

**Type** traits.Bool

**anat\_pipeline**

Instance of AnatomicalPipelineUI class

**Type** Instance(HasTraits)

**anat\_inputs\_checked**

Indicate if anatomical pipeline inputs are available (Default: False)

**Type** traits.Bool

**anat\_outputs\_checked**

Indicate if anatomical pipeline outputs are available (Default: False)

**Type** traits.Bool

**anatomical\_processed**

Indicate if anatomical pipeline was run (Default: False)

**Type** traits.Bool

**dmri\_pipeline**

Instance of DiffusionPipelineUI class

**Type** Instance(HasTraits)

**dmri\_inputs\_checked**

Indicate if diffusion pipeline inputs are available (Default: False)

**Type** traits.Bool

**dmri\_processed**

Indicate if diffusion pipeline was run (Default: False)

**Type** traits.Bool

**fmri\_pipeline**

Instance of fMRIPipelineUI class

**Type** Instance(HasTraits)

**fmri\_inputs\_checked**

Indicate if fMRI pipeline inputs are available (Default: False)

**Type** traits.Bool

**fmri\_processed**

Indicate if fMRI pipeline was run (Default: False)

**Type** traits.Bool

**load\_dataset** (*ui\_info*, *debug=True*)

Function that creates a new CMP\_Project\_InfoUI instance from an existing project.

**Parameters**

- **ui\_info** (*QtView*) – TraitsUI QtView associated with self
- **debug** (*bool*) – If True, print more information for debugging

`cmp.bidsappmanager.project.clean_cache(bids_root)`

Clean cache stored in /tmp.



Target issue related to that a dataset directory is mounted into /tmp and used for caching by java/matlab/matplotlib/xvfb-run in the container image.

**Parameters** `bids_root` (*string*) – BIDS root dataset directory

`cmp.bidsappmanager.project.init_anat_project` (*project\_info*, *is\_new\_project*)

Create and initialize a AnatomicalPipelineUI instance

**Parameters**

- **project\_info** (*CMP\_Project\_InfoUI*) – Instance of *CMP\_Project\_InfoUI* class
- **is\_new\_project** (*bool*) – If True, this is a new project which has been never processed

`cmp.bidsappmanager.project.init_dmri_project` (*project\_info*, *bids\_layout*, *is\_new\_project*, *gui=True*)

Create and initialize a DiffusionPipelineUI instance

**Parameters**

- **project\_info** (*CMP\_Project\_InfoUI*) – Instance of *CMP\_Project\_InfoUI* class
- **bids\_layout** (*bids.BIDSLayout*) – PyBIDS BIDS Layout object describing the BIDS dataset
- **is\_new\_project** (*bool*) – If True, this is a new project which has been never processed
- **gui** (*bool*) – If True, display messages in GUI

`cmp.bidsappmanager.project.init_fmri_project` (*project\_info*, *bids\_layout*, *is\_new\_project*, *gui=True*)

Create and initialize a fMRIPipelineUI instance

**Parameters**

- **project\_info** (*CMP\_Project\_InfoUI*) – Instance of *CMP\_Project\_InfoUI* class
- **bids\_layout** (*bids.BIDSLayout*) – PyBIDS BIDS Layout object describing the BIDS dataset
- **is\_new\_project** (*bool*) – If True, this is a new project which has been never processed
- **gui** (*bool*) – If True, display messgae in GUI

`cmp.bidsappmanager.project.is_tool` (*name*)

Check whether name is on PATH.

`cmp.bidsappmanager.project.refresh_folder` (*derivatives\_directory*, *subject*, *input\_folders*, *session=None*)

Creates (if needed) the folder hierarchy.

**Parameters**

- **derivatives\_directory** (*string*) –
- **subject** (*string*) – Subject label (sub-XX) for which we create the output folder hierarchy
- **session** (*string*) – Subject session label (ses-YY)
- **input\_folders** (*list of string*) – List of folders to create in *derivative\_directory/sub-XX/(ses-YY)* / folder for the given subject

`cmp.bidsappmanager.project.update_anat_last_processed(project_info, pipeline)`  
Update anatomical pipeline processing information

**Parameters**

- **project\_info** (`CMP_Project_InfoUI`) – Instance of `CMP_Project_InfoUI` class
- **pipeline** (`AnatomicalPipelineUI`) – Instance of `AnatomicalPipelineUI`

`cmp.bidsappmanager.project.update_dmri_last_processed(project_info, pipeline)`  
Update diffusion pipeline processing information

**Parameters**

- **project\_info** (`CMP_Project_InfoUI`) – Instance of `CMP_Project_InfoUI` class
- **pipeline** (`DiffusionPipelineUI`) – Instance of `DiffusionPipelineUI`

`cmp.bidsappmanager.project.update_fmri_last_processed(project_info, pipeline)`  
Update functional MRI pipeline processing information

**Parameters**

- **project\_info** (`CMP_Project_InfoUI`) – Instance of `CMP_Project_InfoUI` class
- **pipeline** (`fMRIPipelineUI`) – Instance of `fMRIPipelineUI`

## **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, gui=True*)

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

**Parameters**

- **layout** (*bids.BIDSLayout*) – BIDSLayout object used to query
- **gui** (*Boolean*) – If True display messages and errors in graphical window

**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

**class** `cmp.bidsappmanager.pipelines.anatomical.anatomical.Check_Input_NotificationUI`

Bases: `cmp.pipelines.anatomical.anatomical.Check_Input_Notification`

**cmp.bidsappmanager.pipelines.diffusion package****Submodules****cmp.bidsappmanager.pipelines.diffusion.diffusion module**

Diffusion pipeline UI Class definition.

**class** `cmp.bidsappmanager.pipelines.diffusion.diffusion.Check_Input_NotificationUI`

Bases: `cmp.pipelines.diffusion.diffusion.Check_Input_Notification`

**class** `cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipelineUI` (*project\_info*)

Bases: `cmp.pipelines.diffusion.diffusion.DiffusionPipeline`

Class that extends the `DiffusionPipeline` 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`

**diffusion**

Button to open the window for configuration or quality inspection of the diffusion reconstruction and tractography 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.diffusion.diffusion.DiffusionPipeline`

**check\_input** (*layout*, *gui=True*)

Method that checks if inputs of the diffusion 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 anatomical pipeline are available

**Return type** `bool`

**cmp.bidsappmanager.pipelines.functional package****Submodules****cmp.bidsappmanager.pipelines.functional.eeg module**

EEG pipeline Class definition

**cmp.bidsappmanager.pipelines.functional.fMRI module**

Functional pipeline UI Class definition.

**class** cmp.bidsappmanager.pipelines.functional.fMRI.**Check\_Input\_NotificationUI**  
 Bases: *cmp.pipelines.functional.fMRI.Check\_Input\_Notification*

**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*,  
*out-put\_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.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*,  
*out-*  
*put\_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**

Definition of diffusion reconstruction and tracking config and stage UI classes.

**class** cmp.bidsappmanager.stages.diffusion.diffusion.**DiffusionConfigUI**

Bases: *cmp.stages.diffusion.diffusion.DiffusionConfig*

Class that extends the DiffusionConfig with graphical components.

It includes the graphical components defining the configuration of the diffusion reconstruction and tractography sub-stages.

**traits\_view**

TraitsUI view that displays the Diffusion Stage Parameter Attributes of this class

**Type** traits.ui.View

**See also:**

*cmp.stages.diffusion.diffusion.DiffusionConfig*

**class** cmp.bidsappmanager.stages.diffusion.diffusion.**DiffusionStageUI** (*bids\_dir,*  
*out-*  
*put\_dir*)

Bases: *cmp.stages.diffusion.diffusion.DiffusionStage*

Class that extends the DiffusionStage 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.diffusion.diffusion.DiffusionStage*



**cmp.bidsappmanager.stages.diffusion.reconstruction module**

Definition of diffusion reconstruction config UI classes.

**class** cmp.bidsappmanager.stages.diffusion.reconstruction.**Dipy\_recon\_configUI**

Bases: *cmp.stages.diffusion.reconstruction.Dipy\_recon\_config*

Class that extends the Dipy\_recon\_config with graphical components.

**flip\_table\_axis**

List of axis to flip in the gradient table. Valid values are 'x', 'y', 'z'

**Type** list of string

**traits\_view**

TraitsUI view that displays the attributes of this class, e.g. the parameters for diffusion reconstruction using Dipy

**Type** traits.ui.View

**See also:**

*cmp.stages.diffusion.reconstruction.Dipy\_recon\_config*

**class** cmp.bidsappmanager.stages.diffusion.reconstruction.**MRtrix\_recon\_configUI**

Bases: *cmp.stages.diffusion.reconstruction.MRtrix\_recon\_config*

Class that extends the MRtrix\_recon\_config with graphical components.

**flip\_table\_axis**

List of axis to flip in the gradient table. Valid values are 'x', 'y', 'z'

**Type** list of string

**traits\_view**

TraitsUI view that displays the attributes of this class, e.g. the parameters for diffusion reconstruction using MRtrix

**Type** traits.ui.View

**See also:**

*cmp.stages.diffusion.reconstruction.MRtrix\_recon\_config*

**cmp.bidsappmanager.stages.diffusion.tracking module**

Definition of diffusion tracking config UI classes.

**class** cmp.bidsappmanager.stages.diffusion.tracking.**Dipy\_tracking\_configUI**

Bases: *cmp.stages.diffusion.tracking.Dipy\_tracking\_config*

Class that extends the Dipy\_tracking\_config with graphical components.

**traits\_view**

TraitsUI view that displays the attributes of this class, e.g. the parameters for diffusion tractography using Dipy

**Type** traits.ui.View

**See also:**

*cmp.stages.diffusion.tracking.Dipy\_tracking\_config*

**class** `cmp.bidsappmanager.stages.diffusion.tracking.MRtrix_tracking_configUI`

Bases: `cmp.stages.diffusion.tracking.MRtrix_tracking_config`

Class that extends the `MRtrix_tracking_config` with graphical components.

**traits\_view**

TraitsUI view that displays the attributes of this class, e.g. the parameters for tractography using MRtrix

**Type** `traits.ui.View`

**See also:**

`cmp.stages.diffusion.reconstruction.MRtrix_tracking_config`

## **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*,  
*out-dir*,  
*put\_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.

**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,*  
*bids\_dir,*  
*out-*  
*put\_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*,  
*out\_dir*,  
*put\_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*,  
*out\_dir*,  
*put\_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="",*  
*out-*  
*put\_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.

**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 (*bids\_dir*,  
*out-dir*  
*put\_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.utils module

This modules provides CMTK Utility functions to handle BIDS datasets.

`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', 'freesurfer', 'nipype'])

#### 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 nipy.interfaces import afni as afni
>>> from nipy.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’ millimeters.

**++ 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 nipy.interfaces import afni
>>> despikes = afni.Despike()
>>> despikes.inputs.in_file = 'functional.nii'
>>> despikes.cmdline
'3dDespike -prefix functional_despike functional.nii'
>>> res = despikes.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)
```

(continues on next page)

(continued from previous page)

```
>>> apply_tf.inputs.interpolation = 'NearestNeighbor'
>>> apply_tf.inputs.default_value = 0.0
>>> apply_tf.inputs.out_postfix = "_transformed"
>>> apply_tf.run()
```

**reference\_image** : a filename or object implementing the `os.PathLike` interface transforms : a list of items which are a filename or object implementing the `os.PathLike` interface

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 filename or object implementing the `os.PathLike` interface **interpolation** : 'Linear' or 'NearestNeighbor' or 'CosineWindowedSinc' or 'WelchWindowedSinc' or 'HammingWindowedSinc' or 'LanczosWindowedSinc' or 'MultiLabel' or 'Gaussian' or 'BSpline'

(Nipy **default** value: Linear)

**out\_postfix** [a string] (Nipy **default** value: `_transformed`)

**output\_images** : a list of items which are a filename or object implementing the `os.PathLike` interface

## cmtklib.interfaces.camino module

The Camino module provides Nipy interfaces for Camino functions missing in nipy or modified.

## DTLUTGen

[Link to code](#)

Bases: `nipy.interfaces.base.core.StdOutCommandLine`

Wrapped executable: `dtlutgen`.

Calibrates the PDFs for PICO probabilistic tractography.

This program needs to be run once for every acquisition scheme. It outputs a lookup table that is used by the `dtpicoparams` program to find PICO PDF parameters for an image. The default single tensor LUT contains parameters of the Bingham distribution and is generated by supplying a scheme file and an estimated signal to noise in white matter regions of the ( $q=0$ ) image. The default inversion is linear (inversion index 1).

Advanced users can control several options, including the extent and resolution of the LUT, the inversion index, and the type of PDF. See `dtlutgen(1)` for details.

## Example

```
>>> import cmtklib.interfaces.camino as cmon
>>> dtl = cmon.DTLUTGen()
>>> dtl.inputs.snr = 16
>>> dtl.inputs.scheme_file = 'A.scheme'
>>> dtl.run()
```

**scheme\_file** [a pathlike object or string representing a file] The scheme file of the images to be processed using this LUT. Maps to a command-line argument: `-schemefile %s` (position: 2).

- acg** [a boolean] Compute a LUT for the ACG PDF. Maps to a command-line argument: `-acg`.
- args** [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.
- bingham** [a boolean] Compute a LUT for the Bingham PDF. This is the default. Maps to a command-line argument: `-bingham`.
- cross** [a float] The angle in degrees between the principal directions of the two tensors. Maps to a command-line argument: `-cross %d`.
- 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: `{}`)
- frange** [a list of from 2 to 2 items which are a float] Index to two-tensor LUTs. This is the fractional anisotropy of the two tensors. The default is 0.3 to 0.94. Maps to a command-line argument: `-frange %s (position: 1)`.
- inversion** [an integer] Index of the inversion to use. The default is 1 (linear single tensor inversion). Maps to a command-line argument: `-inversion %d`.
- lrange** [a list of from 2 to 2 items which are a float] Index to one-tensor LUTs. This is the ratio  $L1/L3$  and  $L2/L3$ . The LUT is square, with half the values calculated (because  $L2/L3$  cannot be less than  $L1/L3$  by definition). The minimum must be  $\geq 1$ . For comparison, a ratio  $L1/L3 = 10$  with  $L2/L3 = 1$  corresponds to an FA of 0.891, and  $L1/L3 = 15$  with  $L2/L3 = 1$  corresponds to an FA of 0.929. The default range is 1 to 10. Maps to a command-line argument: `-lrange %s (position: 1)`.
- out\_file** [a pathlike object or string representing a file] Maps to a command-line argument: `> %s (position: -1)`.
- samples** [an integer] The number of synthetic measurements to generate at each point in the LUT. The default is 2000. Maps to a command-line argument: `-samples %d`.
- snr** [a float] The signal to noise ratio of the unweighted ( $q = 0$ ) measurements. This should match the SNR (in white matter) of the images that the LUTs are used with. Maps to a command-line argument: `-snr %d`.
- step** [a float] Distance between points in the LUT. For example, if lrange is 1 to 10 and the step is 0.1, LUT entries will be computed at  $L1/L3 = 1, 1.1, 1.2 \dots 10.0$  and at  $L2/L3 = 1.0, 1.1 \dots L1/L3$ . For single tensor LUTs, the default step is 0.2, for two-tensor LUTs it is 0.02. Maps to a command-line argument: `-step %d`.
- trace** [a float] Trace of the diffusion tensor(s) used in the test function in the LUT generation. The default is  $2100E-12 \text{ m}^2 \text{ s}^{-1}$ . Maps to a command-line argument: `-trace %f`.
- watson** [a boolean] Compute a LUT for the Watson PDF. Maps to a command-line argument: `-watson`.
- dtLUT** [a pathlike object or string representing an existing file] Lookup Table.

## PicoPDFs

### [Link to code](#)

Bases: `nipype.interfaces.base.core.StdOutCommandLine`

Wrapped executable: `picopdfs`.

Constructs a spherical PDF in each voxel for probabilistic tractography.

## Example

```
>>> import cmtklib.interfaces.camino as cmon
>>> pdf = cmon.PicoPDFs()
>>> pdf.inputs.inputmodel = 'dt'
>>> pdf.inputs.luts = 'lut_file'
>>> pdf.inputs.in_file = 'voxel-order_data.Bfloat'
>>> pdf.run()
```

**in\_file** [a pathlike object or string representing an existing file] Voxel-order data filename. Maps to a command-line argument: `< %s` (position: 1).

**args** [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

**directmap** [a boolean]

**Only applicable when using pds as the inputmodel** Use direct mapping between the eigenvalues and the distribution parameters instead of the log of the eigenvalues.

Maps to a command-line argument: `-directmap`.

**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: `{}`)

**inputmodel** ['dt' or 'multitensor' or 'pds'] Input model type. Maps to a command-line argument: `-inputmodel %s` (position: 2). (Nipype **default** value: `dt`)

**luts** [a list of items which are a pathlike object or string representing an existing file] Files containing the lookup tables. For tensor data, one lut must be specified for each type of inversion used in the image (one-tensor, two-tensor, three-tensor). For pds, the number of LUTs must match -numpds (it is acceptable to use the same LUT several times - see example, above). These LUTs may be generated with `dtlutgen`. Maps to a command-line argument: `-luts %s` (position: 3).

**maxcomponents** [an integer] The maximum number of tensor components in a voxel (default 2) for multitensor data. Currently, only the default is supported, but future releases may allow the input of three-tensor data using this option. Maps to a command-line argument: `-maxcomponents %d`.

**numpds** [an integer] The maximum number of PDs in a voxel (default 3) for PD data. This option determines the size of the input and output voxels. This means that the data file may be large enough to accommodate three or more PDs, but does not mean that any of the voxels are classified as containing three or more PDs. Maps to a command-line argument: `-numpds %d`.

**out\_file** [a pathlike object or string representing a file] Maps to a command-line argument: `> %s` (position: -1).

**pdf** ['watson' or 'bingham' or 'acg']

Specifies the PDF to use. There are three choices: `watson` - The Watson distribution. This distribution is rotationally symmetric. `bingham` - The Bingham distribution, which allows elliptical probability density contours. `acg` - The Angular Central Gaussian distribution, which also allows elliptical probability density contours.

Maps to a command-line argument: `-pdf %s` (position: 4). (Nipype **default** value: `watson`)

**pdfs** [a pathlike object or string representing an existing file] Path/name of 4D volume in voxel order.

## Voxel2Image

[Link to code](#)

Bases: `nipy.interfaces.base.core.StdOutCommandLine`

Wrapped executable: `voxel2image`.

Converts voxel order image to NIFTI / MHA files.

Converts voxel-order data to scanner-order data in a supported image format.

### Examples

```
>>> import cmtklib.interfaces.camino as cmon
>>> vox2img = cmon.Voxel2Image()
>>> vox2img.inputs.in_file = 'fa.img'
>>> vox2img.run()
```

**header\_file** [a pathlike object or string representing an existing file] File with desired format. Maps to a command-line argument: `-header %s` (position: 2).

**in\_file** [a pathlike object or string representing an existing file] Image in camino format. Maps to a command-line argument: `-inputfile %s` (position: 1).

**args** [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

**components** [an integer] Number of components in inputfile. Maps to a command-line argument: `-components %d`.

**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: `{}`)

**gzip** [a boolean] Compress output image. Maps to a command-line argument: `-gzip`.

**out\_file** [a pathlike object or string representing a file] Maps to a command-line argument: `> %s` (position: -1).

**out\_type** ['float' or 'char' or 'short' or 'int' or 'long' or 'double'] "i.e. Bfloat". Can be "char", "short", "int", "long", "float" or "double". Maps to a command-line argument: `-outputdatatype %s`. (Nipype **default** value: `float`)

**output\_root** [a string] Maps to a command-line argument: `-outputroot %s` (position: -1). (Nipype **default** value: `converted`)

**image\_file** [a pathlike object or string representing an existing file] Path/name to converted file.

### `cmtklib.interfaces.camino2trackvis` module

Provides interfaces for functions provided by Camino-Tackvis missing in nipype or modified.

## Camino2Trackvis

[Link to code](#)

Bases: `nipype.interfaces.base.core.CommandLine`

Wrapped executable: `camino_to_trackvis`.

Wraps `camino_to_trackvis` from Camino-Trackvis.

Convert files from camino .Bfloat format to trackvis .trk format.

### Example

```
>>> import cmtklib.interfaces.camino2trackvis as cam2trk
>>> c2t = cam2trk.Camino2Trackvis()
>>> c2t.inputs.in_file = 'data.Bfloat'
>>> c2t.inputs.out_file = 'streamlines.trk'
>>> c2t.inputs.min_length = 30
>>> c2t.inputs.data_dims = [128, 104, 64]
>>> c2t.inputs.voxel_dims = [2.0, 2.0, 2.0]
>>> c2t.inputs.voxel_order = 'LAS'
>>> c2t.run()
```

**in\_file** [a pathlike object or string representing an existing file] The input .Bfloat (camino) file. Maps to a command-line argument: `-i %s` (position: 1).

**args** [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

**data\_dims** [a list of from 3 to 3 items which are an integer] Three comma-separated integers giving the number of voxels along each dimension of the source scans. Maps to a command-line argument: `-d %s` (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: `{}`)

**min\_length** [a float] The minimum length of tracts to output. Maps to a command-line argument: `-l %d` (position: 3).

**nifti\_file** [a pathlike object or string representing an existing file] Read coordinate system from a NIFTI file. Maps to a command-line argument: `--nifti %s` (position: 7).

**out\_file** [a pathlike object or string representing a file] The filename to which to write the .trk (trackvis) file. Maps to a command-line argument: `-o %s` (position: 2).

**phys\_coords** [a boolean] Treat the input tract points as physical coordinates (relevant for the updated camino track command). Maps to a command-line argument: `--phys-coords` (position: 8).

**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: `-x %s` (position: 5).

**voxel\_order** [a pathlike object or string representing a file] Set the order in which various directions were stored. Specify with three letters consisting of one each from the pairs LR, AP, and SI. These stand for Left-Right, Anterior-Posterior, and Superior-Inferior. Whichever is specified in each position will be the direction of increasing order. Read coordinate system from a NIFTI file. Maps to a command-line argument: `--voxel-order %s` (position: 6).

**trackvis** [a pathlike object or string representing an existing file] The filename to which to write the .trk (trackvis) file.

## cmtklib.interfaces.diffusion\_toolkit module

The Diffusion Toolkit module provides Nipype interfaces for the Diffusion Toolkit missing in nipype or modified.

### DTIRecon

[Link to code](#)

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: dti\_recon.

Use dti\_recon to generate tensors and other maps.

---

**Note:** Not used anymore by CMP3

---

**DWI** [a pathlike object or string representing an existing file] Input diffusion volume. Maps to a command-line argument: %s (position: 1).

**gradient\_matrix** [a pathlike object or string representing an existing file] Specify gradient matrix to use. required. Maps to a command-line argument: -gm %s (position: 3).

**args** [a string] Additional parameters to the command. Maps to a command-line argument: %s.

**b0\_threshold** [a float]

**Program will use b0 image with the given threshold to mask out high background of fa/adc maps.** by default it will calculate threshold automatically. but if it failed, you need to set it manually.

Maps to a command-line argument: -b0\_th.

**b\_value** [an integer] Set b value or maximum b value for multi-bvalue data. default is 1000. Maps to a command-line argument: -b %d.

**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: { })

**image\_orientation\_vectors** [a list of from 6 to 6 items which are a float]

**Specify image orientation vectors. if just one argument given,** will treat it as filename and read the orientation vectors from the file. if 6 arguments are given, will treat them as 6 float numbers and construct the 1st and 2nd vector and calculate the 3rd one automatically. this information will be used to determine image orientation, as well as to adjust gradient vectors with oblique angle when.

Maps to a command-line argument: -iop %f.

**multiple\_b\_values** [a boolean]

**If 'MultiBvalue' is 'true' or 1,** it will either use the bvalues specified as the 4th component of each gradient vector, or use max b value scaled by the magnitude of the vector.

Maps to a command-line argument: %d (position: 4).

**n\_averages** [an integer] Number of averages. Maps to a command-line argument: -nex %s.

**number\_of\_b0** [an integer]

**Number of repeated b0 images on top. default is 1. the program** assumes b0 images are on top.

Maps to a command-line argument: -b0 %d.



**oblique\_correction** [a boolean]

**When oblique angle(s) applied, some SIEMENS dti protocols do not** adjust gradient accordingly, thus it requires adjustment for correct diffusion tensor calculation.

Maps to a command-line argument: `-oc`.

**out\_prefix** [a string] Output file prefix. Maps to a command-line argument: `%s` (position: 2). (Nipype **default** value: `dti`)

**output\_type** ['nii' or 'analyze' or 'nii.gz'] Output file type. Maps to a command-line argument: `-ot %s`. (Nipype **default** value: `nii`)

ADC : a pathlike object or string representing an existing file  
 B0 : a pathlike object or string representing an existing file  
 DWI : a pathlike object or string representing an existing file  
 FA : a pathlike object or string representing an existing file  
 FA\_color : a pathlike object or string representing an existing file  
 L1 : a pathlike object or string representing an existing file  
 L2 : a pathlike object or string representing an existing file  
 L3 : a pathlike object or string representing an existing file  
 V1 : a pathlike object or string representing an existing file  
 V2 : a pathlike object or string representing an existing file  
 V3 : a pathlike object or string representing an existing file  
 exp : a pathlike object or string representing an existing file  
 tensor : a pathlike object or string representing an existing file

## DiffUnpack

[Link to code](#)

Bases: `nipype.interfaces.base.core.CommandLine`

Wrapped executable: `diff_unpack`.

Use `diff_unpack` to convert dicom files to multiple formats.

## Examples

```
>>> convert = DiffUnpack()
>>> convert.inputs.input_dicom = '/path/to/sub-01_dwi.dcm'
>>> convert.inputs.out_prefix = 'output'
>>> convert.inputs.output_type = 'nii.gz'
>>> convert.run()
```

**input\_dicom** [a pathlike object or string representing an existing file] Input dicom file. 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\_prefix** [a string] Output file prefix. Maps to a command-line argument: `%s` (position: 2). (Nipype **default** value: `output`)

**output\_type** ['nii' or 'analyze' or 'nii.gz'] Output file type. Maps to a command-line argument: `-ot %s`. (Nipype **default** value: `nii`)

**split** [a boolean]

**Instead of saving everything in one big multi-timepoint 4D image, split it into separate files, one timepoint per file.**

Maps to a command-line argument: `-split`.

**converted\_files** [a list of items which are any value] Converted files.

## HARDIMat

[Link to code](#)

Bases: `nipype.interfaces.base.core.CommandLine`

Wrapped executable: `hardi_mat`.

Use `hardi_mat` to calculate a reconstruction matrix from a gradient table.

## Examples

```
>>> hardi_mat = HARDIMat()
>>> hardi_mat.inputs.bvecs = 'sub-01_dwi.bvec'
>>> hardi_mat.inputs.bvals = 'sub-01_dwi.bval'
>>> hardi_mat.inputs.gradient_table = 'sub-01_grad.txt'
>>> hardi_mat.inputs.out_file = 'recon_mat.dat'
>>> hardi_mat.inputs.order = 8
>>> hardi_mat.inputs.reference_file = 'sub-01_dwi.nii.gz'
>>> hardi_mat.run()
```

**args** [a string] Additional parameters to the command. Maps to a command-line argument: `%s`.

**bvals** [a pathlike object or string representing an existing file] B values file.

**bvecs** [a pathlike object or string representing an existing file] B vectors file. Maps to a command-line argument: `%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: `{}`)

**gradient\_table** [a pathlike object or string representing an existing file] Input gradient table. Maps to a command-line argument: `%s` (position: 1).

**image\_info** [a pathlike object or string representing an existing file]

**Specify image information file. the image info file is generated** from original dicom image by `diff_unpack` program and contains image orientation and other information needed for reconstruction and tracking. by default will look into the image folder for `.info` file.

Maps to a command-line argument: `-info %s`.

**image\_orientation\_vectors** [a list of from 6 to 6 items which are a float]

**Specify image orientation vectors. if just one argument given,** will treat it as filename and read the orientation vectors from the file. if 6 arguments are given, will treat them as 6 float numbers and construct the 1st and 2nd vector and calculate the 3rd one automatically. This information will be used to determine image orientation, as well as to adjust gradient vectors with oblique angle when.

Maps to a command-line argument: `-iop %f`.

**oblique\_correction** [a boolean]

**When oblique angle(s) applied, some SIEMENS dti** protocols do not adjust gradient accordingly, thus it requires adjustment for correct diffusion tensor calculation.

Maps to a command-line argument: `-oc`.

**odf\_file** [a pathlike object or string representing an existing file]

**Filename that contains the reconstruction points on a HEMI-sphere.** use the pre-set 181 points by default.

Maps to a command-line argument: `-odf %s`.

**order** [an integer]

**Maximum order of spherical harmonics. must be even number. default** is 4.

**out\_file** [a pathlike object or string representing a file] Output matrix file. Maps to a command-line argument: `%s` (position: 2). (Nipype **default** value: `recon_mat.dat`)

**reference\_file** [a pathlike object or string representing an existing file]

**Provide a dicom or nifti image as the reference for the program to** figure out the image orientation information. if no such info was found in the given image header, the next 5 options -info, etc., will be used if provided. if image orientation info can be found in the given reference, all other 5 image orientation options will be IGNORED.

Maps to a command-line argument: `-ref %s`.

**out\_file** [a pathlike object or string representing an existing file] Output matrix file.

## cmtklib.interfaces.dipy module

The Dipy module provides Nipype interfaces to the algorithms in dipy.

## CSD

[Link to code](#)

Bases: `nipype.interfaces.dipy.base.DipyDiffusionInterface`

Uses CSD [[Tournier2007](#)] to generate the fODF of DWIs.

The interface uses `dipy`, as explained in [dipy's CSD example](#).

## References

## Example

```
>>> from cmtklib.interfaces.dipy import CSD
>>> csd = CSD()
>>> csd.inputs.in_file = '4d_dwi.nii'
>>> csd.inputs.in_bval = 'bvals'
>>> csd.inputs.in_bvec = 'bvecs'
>>> res = csd.run()
```

**in\_bval** [a pathlike object or string representing an existing file] Input b-values table.

**in\_bvec** [a pathlike object or string representing an existing file] Input b-vectors table.

**in\_file** [a pathlike object or string representing an existing file] Input diffusion data.

**b0\_thres** [an integer] B0 threshold. (Nipype **default** value: 700)

**fa\_thresh** [a float] FA threshold used for response estimation. (Nipype **default** value: 0.7)

**in\_mask** [a pathlike object or string representing an existing file] Input mask in which compute tensors.

**out\_fods** [a pathlike object or string representing a file] FODFs output file name.

**out\_prefix** [a string] Output prefix for file names.

**out\_shm\_coeff** [a pathlike object or string representing a file] Spherical Harmonics Coefficients output file name.

**response** [a pathlike object or string representing an existing file] Single fiber estimated response.

**save\_fods** [a boolean] Save fODFs in file. (Nipype **default** value: `True`)

**save\_shm\_coeff** [a boolean] Save Spherical Harmonics Coefficients in file. (Nipype **default** value: `True`)

**sh\_order** [an integer] Maximal shperical harmonics order. (Nipype **default** value: 8)

**tracking\_processing\_tool** : 'mrtix' or 'dipy'

**model** [a pathlike object or string representing a file] Python pickled object of the CSD model fitted.

**out\_fods** [a pathlike object or string representing a file] FODFs output file name.

**out\_shm\_coeff** [a pathlike object or string representing a file] Spherical Harmonics Coefficients output file name.

## DTIEstimateResponseSH

[Link to code](#)

Bases: `nipype.interfaces.dipy.base.DipyDiffusionInterface`

Uses dipy to compute the single fiber response to be used by spherical deconvolution methods.

The single fiber response is computed in a similar way to MRTrix's command `estimate_response`.

### Example

```
>>> from cmtklib.interfaces.dipy import DTIEstimateResponseSH
>>> dti = DTIEstimateResponseSH()
>>> dti.inputs.in_file = '4d_dwi.nii'
>>> dti.inputs.in_bval = 'bvals'
>>> dti.inputs.in_bvec = 'bvecs'
>>> res = dti.run()
```

**in\_bval** [a pathlike object or string representing an existing file] Input b-values table.

**in\_bvec** [a pathlike object or string representing an existing file] Input b-vectors table.

**in\_file** [a pathlike object or string representing an existing file] Input diffusion data.

**auto** [a boolean] Use the `auto_response` estimator from dipy. Mutually **exclusive** with `inputs_recursive`.

**b0\_thres** [an integer] B0 threshold. (Nipype **default** value: 700)

**fa\_thresh** [a float] FA threshold. (Nipype **default** value: 0.7)

**in\_mask** [a pathlike object or string representing an existing file] Input mask in which we find single fibers.

**out\_mask** [a pathlike object or string representing a file] Computed wm mask. (Nipype **default** value: `wm_mask.nii.gz`)

**out\_prefix** [a string] Output prefix for file names.

**recursive** [a boolean] Use the recursive response estimator from dipy. Mutually **exclusive** with inputs: `auto`.

**response** [a pathlike object or string representing a file] The output response file. (Nipype **default** value: `response.txt`)

**roi\_radius** [an integer] ROI radius to be used in `auto_response`. (Nipype **default** value: 10)

`ad_file` : a pathlike object or string representing an existing file `dti_model` : a pathlike object or string representing an existing file

DTI model object.

`fa_file` : a pathlike object or string representing an existing file `md_file` : a pathlike object or string representing an existing file `out_mask` : a pathlike object or string representing an existing file

Output wm mask.

`rd_file` : a pathlike object or string representing an existing file `response` : a pathlike object or string representing an existing file

The response file.

## DirectionGetterTractography

[Link to code](#)

Bases: `nipype.interfaces.dipy.base.DipyBaseInterface`

Streamline tractography using Dipy Deterministic Maximum Direction Getter.

### Example

```
>>> from cmtklib.interfaces import dipy as ndp
>>> track = ndp.DirectionGetterTractography()
>>> track.inputs.in_file = '4d_dwi.nii'
>>> track.inputs.in_model = 'model.pklz'
>>> track.inputs.tracking_mask = 'dilated_wm_mask.nii'
>>> res = track.run()
```

**fa\_thresh** [a float] FA threshold to build the tissue classifier. (Nipype **default** value: 0.2)

**in\_fa** [a pathlike object or string representing an existing file] Input FA.

**in\_file** [a pathlike object or string representing an existing file] Input diffusion data.

**in\_model** [a pathlike object or string representing an existing file] Input f/d-ODF model extracted from.

**max\_angle** [a float] Maximum angle. (Nipype **default** value: 25.0)

**multiprocess** [a boolean] Use multiprocessing. (Nipype **default** value: `True`)

**num\_seeds** [an integer] Desired number of tracks in tractography. (Nipype **default** value: 10000)

**save\_seeds** [a boolean] Save seeding voxels coordinates. (Nipype **default** value: `False`)

**seed\_mask** [a list of items which are a pathlike object or string representing an existing file] ROI files registered to diffusion space.

**step\_size** [a float] Step size. (Nipype **default** value: 0.5)

**tracking\_mask** [a pathlike object or string representing an existing file] Input mask within which perform tracking.

**algo** ['deterministic' or 'probabilistic'] Use either deterministic maximum (default) or probabilistic direction getter tractography. (Nipype **default** value: `deterministic`)

**fod\_file** [a pathlike object or string representing an existing file] Input fod file (if SHORE).

**gmwmi\_file** [a pathlike object or string representing an existing file] Input Gray Matter / White Matter interface image.

**in\_partial\_volume\_files** [a list of items which are a pathlike object or string representing an existing file] Partial volume estimation result files (required if performing ACT).

**out\_prefix** [a string] Output prefix for file names.

**recon\_model** ['CSD' or 'SHORE'] Use either fODFs from CSD (default) or SHORE models. (Nipype **default** value: `CSD`)

**recon\_order** : an integer **seed\_density** : a float

Density of seeds. (Nipype **default** value: 1)

**seed\_from\_gmwmi** [a boolean] Seed from the Gray Matter / White Matter interface.

**use\_act** [a boolean] Use FAST for partial volume estimation and Anatomically-Constrained Tractography (ACT) tissue classifier.

**out\_seeds** [a pathlike object or string representing a file] File containing the (N,3) *voxel* coordinates used in seeding.

**streamlines** [a pathlike object or string representing a file] Numpy array of streamlines.

**tracks** [a pathlike object or string representing a file] TrackVis file containing extracted streamlines.

**tracks2** [a pathlike object or string representing a file] TrackVis file containing extracted streamlines.

**tracks3** [a pathlike object or string representing a file] TrackVis file containing extracted streamlines.

## MAPMRI

[Link to code](#)

Bases: `nipype.interfaces.dipy.base.DipyDiffusionInterface`

Computes the MAP MRI model.

for reference on the settings

## Example

```
>>> from cmtklib.interfaces.dipy import MAPMRI
>>> mapmri = MAPMRI()
>>> mapmri.inputs.in_file = '4d_dwi.nii'
>>> mapmri.inputs.in_bval = 'bvals'
>>> mapmri.inputs.in_bvec = 'bvecs'
>>> res = mapmri.run()
```

**big\_delta** [a float] Small data for gradient table.

**in\_bval** [a pathlike object or string representing an existing file] Input b-values table.

**in\_bvec** [a pathlike object or string representing an existing file] Input b-vectors table.

**in\_file** [a pathlike object or string representing an existing file] Input diffusion data.

**small\_delta** [a float] Small data for gradient table.

**b0\_thres** [an integer] B0 threshold. (Nipype **default** value: 700)

**laplacian\_regularization** [a boolean] Apply laplacian regularization. (Nipype **default** value: True)

**laplacian\_weighting** [a float] Regularization weight. (Nipype **default** value: 0.05)

**out\_prefix** [a string] Output prefix for file names.

**positivity\_constraint** [a boolean] Apply positivity constraint. (Nipype **default** value: True)

**radial\_order** [an integer] Maximal shperical harmonics order. (Nipype **default** value: 8)

**model** [a pathlike object or string representing a file] Python pickled object of the MAP-MRI model fitted.

**msd\_file** [a pathlike object or string representing a file] Msd output file name.

**ng\_file** [a pathlike object or string representing a file] Ng output file name.

**ng\_para\_file** [a pathlike object or string representing a file] Ng parallel output file name.

**ng\_perp\_file** [a pathlike object or string representing a file] Ng perpendicular output file name.

**qiv\_file** [a pathlike object or string representing a file] Qiv output file name.

**rtap\_file** [a pathlike object or string representing a file] Rtap output file name.

**rtop\_file** [a pathlike object or string representing a file] Rtop output file name.

**rtpv\_file** [a pathlike object or string representing a file] Rtpv output file name.

## SHORE

[Link to code](#)

Bases: nipype.interfaces.dipy.base.DipyDiffusionInterface

Uses SHORE [Merlet13] to generate the fODF of DWIs.

The interface uses `dipy`, as explained in [dipy's SHORE example](#).

## References

“Continuous diffusion signal, EAP and ODF estimation via Compressive Sensing in diffusion MRI”

## Example

```
>>> from cmtklib.interfaces.dipy import SHORE
>>> asm = SHORE(radial_order=6, zeta=700, lambda_n=1e-8, lambda_l=1e-8)
>>> asm.inputs.in_file = '4d_dwi.nii'
>>> asm.inputs.in_bval = 'bvals'
>>> asm.inputs.in_bvec = 'bvecs'
>>> res = asm.run()
```

**in\_bval** [a pathlike object or string representing an existing file] Input b-values table.

**in\_bvec** [a pathlike object or string representing an existing file] Input b-vectors table.

**in\_file** [a pathlike object or string representing an existing file] Input diffusion data.

**b0\_thres** [an integer] B0 threshold. (Nipype **default** value: 700)

**constrain\_e0** [a boolean] Constrain the optimization such that  $E(0) = 1$ . (Nipype **default** value: `False`)

**in\_mask** [a pathlike object or string representing an existing file] Input mask in which compute SHORE solution.

**lambda\_l** [a float] Angular regularisation constant. (Nipype **default** value: `1e-08`)

**lambda\_n** [a float] Radial regularisation constant. (Nipype **default** value: `1e-08`)

**out\_prefix** [a string] Output prefix for file names.

**positive\_constraint** [a boolean] Constrain the optimization such that  $E(0) = 1$ . (Nipype **default** value: `False`)

**radial\_order** [an integer] Even number that represents the order of the basis. (Nipype **default** value: 6)

**response** [a pathlike object or string representing an existing file] Single fiber estimated response.

**tau** [a float] Diffusion time. By default the value that makes  $q$  equal to the square root of the b-value.

**tracking\_processing\_tool** : ‘mrtrix’ or ‘dipy’ **zeta** : an integer

Scale factor. (Nipype **default** value: 700)

**GFA** [a pathlike object or string representing a file] Generalized Fractional Anisotropy output file name.

**MSD** [a pathlike object or string representing a file] Mean Square Displacement output file name.

**RTOP** [a pathlike object or string representing a file] Return To Origin Probability output file name.

**dodf** [a pathlike object or string representing a file] Fiber Orientation Distribution Function output file name.

**fodf** [a pathlike object or string representing a file] Fiber Orientation Distribution Function output file name.

**model** [a pathlike object or string representing a file] Python pickled object of the SHORE model fitted.



## TensorInformedEudXTractography

[Link to code](#)

Bases: `nipy.interfaces.dipy.base.DipyBaseInterface`

Streamline tractography using Dipy Deterministic Maximum Direction Getter.

### Example

```
>>> from cmtklib.interfaces import dipy as ndp
>>> track = ndp.TensorInformedEudXTractography()
>>> track.inputs.in_file = '4d_dwi.nii'
>>> track.inputs.in_model = 'model.pklz'
>>> track.inputs.tracking_mask = 'dilated_wm_mask.nii'
>>> res = track.run()
```

**fa\_thresh** [a float] FA threshold to build the tissue classifier. (Nipype **default** value: 0.2)

**in\_fa** [a pathlike object or string representing an existing file] Input FA.

**in\_file** [a pathlike object or string representing an existing file] Input diffusion data.

**in\_model** [a pathlike object or string representing an existing file] Input Tensor model extracted from.

**max\_angle** [a float] Maximum angle. (Nipype **default** value: 25.0)

**multiprocess** [a boolean] Use multiprocessing. (Nipype **default** value: True)

**num\_seeds** [an integer] Desired number of tracks in tractography. (Nipype **default** value: 10000)

**save\_seeds** [a boolean] Save seeding voxels coordinates. (Nipype **default** value: False)

**seed\_mask** [a list of items which are a pathlike object or string representing an existing file] ROI files registered to diffusion space.

**step\_size** [a float] Step size. (Nipype **default** value: 0.5)

**tracking\_mask** [a pathlike object or string representing an existing file] Input mask within which perform tracking.

**out\_prefix** [a string] Output prefix for file names.

**out\_seeds** [a pathlike object or string representing a file] File containing the (N,3) *voxel* coordinates used in seeding.

**tracks** [a pathlike object or string representing a file] TrackVis file containing extracted streamlines.

## cmtklib.interfaces.freesurfer module

The FreeSurfer module provides Nipype interfaces for FreeSurfer tools missing in nipype or modified.

## BBRegister

[Link to code](#)

Bases: `nipy.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: `nipy.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:** Note 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.

**out\_files** [a list of items which are a pathlike object or string representing a file] Transformed files.

### BinaryThreshold

[Link to code](#)

Bases: `nipy.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 cmcklib.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 cmcklib.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 MRTrx format.

`index` : a pathlike object or string representing an existing file

## Eddy

[Link to code](#)

Bases: `nipy.interfaces.fsl.base.FSLCommand`

Wrapped executable: `eddy`.

Performs eddy current distortion 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: `nipy.interfaces.fsl.base.FSLCommand`

Wrapped executable: `eddy_openmp`.

Performs eddy current distortion 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: `nipy.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: `nipy.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: `nipy.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.

## **cmrklb.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 cmrklb.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 header of a Nifti image.

#### **Examples**

```
>>> from cmrklb.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

## 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: `nipy.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: `nipy.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: `nipy.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]  $\text{Im} \{Y(2,2)\}$
- [2]  $\text{Im} \{Y(2,1)\}$
- [3]  $Y(2,0)$
- [4]  $\text{Re} \{Y(2,1)\}$
- [5]  $\text{Re} \{Y(2,2)\}$
- [6]  $\text{Im} \{Y(4,4)\}$
- [7]  $\text{Im} \{Y(4,3)\}$

### 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<sup>2</sup>). 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: `nipy.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<sup>2</sup>). 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: `nipy.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: `nipy.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_WMPProb.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<sup>2</sup>). 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 cmcklib.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
>>> genWGMGI = cmp_mrt.Generate5tt()
>>> genWGMGI.inputs.in_file = '5tt.mif'
>>> genWGMGI.inputs.out_file = 'gmwmi.mif'
>>> genWGMGI.cmdline
'5tt2gmwmi 5tt.mif gmwmi.mif'
>>> genWGMGI.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<sup>2</sup>). 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: `nipy.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: *MRTrx3Base*

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: `nipy.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 selected and 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`-value in units (1000 s/mm<sup>2</sup>). 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: `nipy.interfaces.base.core.CommandLine`

Wrapped executable: `tensor2metric`.

Generates a map of the major eigenvectors of the tensors in each voxel using `tensor2metric`.

### Example

```
>>> import cmcklib.interfaces.mrt 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.

## Submodules

### cmcklib.config module

Module that defines CMTK Config file function.

`cmcklib.config.anat_load_config_ini(pipeline, config_path)`

Load the configuration file of an anatomical pipeline.

#### Parameters

- **pipeline** (`Instance(cmp.pipelines.anatomical.anatomical.AnatomicalPipeline)`) – Instance of AnatomicalPipeline
- **config\_path** (`string`) – Path of the configuration file

`cmcklib.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 configuration file

`cmtklib.config.dmri_load_config_ini (pipeline, config_path)`

Load 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 INI configuration file

`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 INI configuration file

`cmtklib.config.fmri_load_config_ini (pipeline, config_path)`

Load 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 INI 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 INI 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 INI configuration file

`cmtklib.config.get_anat_process_detail_ini` (*project\_info, section, detail*)

Get the value for a parameter key (detail) in the stage section of the anatomical INI 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_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.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_dmri_process_detail_ini` (*project\_info, section, detail*)

Get the value for a parameter key (detail) in the stage section of the diffusion INI 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_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.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_ini` (*project\_info*, *section*, *detail*)

Get the value for a parameter key (detail) in the stage section of the fMRI INI 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.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_process_detail_ini` (*project\_info*, *section*, *detail*)

Get the value for a parameter key (detail) in the Global section of the INI config file.

**Parameters**

- **project\_info** (*Instance* (`cmp.project.CMP_Project_Info`)) – Instance of `cmp.project.CMP_Project_Info` class
- **section** (*string*) – Stage 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.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.connectome module

Module that defines CMTK functions and Nipype interfaces for connectome mapping.

### CMTK\_cmat

[Link to code](#)

Bases: `nipype.interfaces.base.core.BaseInterface`

Creates the structural connectivity matrices for a given parcellation scheme.

### Examples

```
>>> from cmtklib.connectome import CMTK_cmat
>>> cmat = CMTK_cmat()
>>> 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',
>>>                               '/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** ['Lausanne2008' or 'Lausanne2018' or 'NativeFreesurfer' or 'Custom'] Parcellation scheme. (Nipype **default** value: Lausanne2008)

**probtrackx** [a boolean] MUST be set to True if probtrackx was used (Not used anymore in CMP3).

**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.

```
cmtklib.connectome.cmat(intrk, roi_volumes, roi_graphmls, parcellation_scheme, compute_curvature=True, additional_maps={}, output_types=['gPickle',  
                                atlas_info={})
```

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', 'Lausanne2008', 'Lausanne2018']*) –
- **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*)

`cmctklib.connectome.group_analysis_sconn` (*output\_dir, subjects\_to\_be\_analyzed*)  
Perform group level analysis of structural connectivity matrices.

### rsfmri\_conmat

[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 pair.

#### Examples

```
>>> from cmctklib.connectome import rsfmri_conmat
>>> cmat = rsfmri_conmat()
>>> 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** ['Lausanne2008' or 'Lausanne2018' or 'NativeFreesurfer' or 'Custom'] Parcellation scheme. (Nipype **default** value: Lausanne2008)

**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.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.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: `nipy.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.

## Make\_Mrtrix\_Seeds

[Link to code](#)

Bases: `nipype.interfaces.base.core.BaseInterface`

Creates seeding ROIs by intersecting dilated ROIs with WM mask for `mrtrix`.

### Examples

```
>>> from cmtklib.diffusion import Make_Mrtrix_Seeds
>>> make_mrtrix_seeds = Make_Mrtrix_Seeds()
>>> make_mrtrix_seeds.inputs.ROI_files = ['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']
>>> make_mrtrix_seeds.inputs.WM_file = 'sub-01_space-DWI_label-WM_dseg.nii.gz
↳'
>>> make_mrtrix_seeds.run()
```

**WM\_file** [a pathlike object or string representing a file] WM mask file registered to diffusion space.

**ROI\_files** [a list of items which are a pathlike object or string representing an existing file] ROI files registered to diffusion space.

**seed\_files** [a list of items which are a pathlike object or string representing an existing file] Seed files for probabilistic tractography.

```
Make_Mrtrix_Seeds.ROI_idx = []
```

```
Make_Mrtrix_Seeds.base_name = ''
```

## Make\_Seeds

[Link to code](#)

Bases: `nipy.interfaces.base.core.BaseInterface`

Creates seeding ROIs by intersecting dilated ROIs with WM mask for Dipy.

## Examples

```
>>> from cmtklib.diffusion import Make_Seeds
>>> make_dipy_seeds = Make_Seeds()
>>> make_dipy_seeds.inputs.ROI_files = ['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']
>>> make_dipy_seeds.inputs.WM_file = 'sub-01_space-DWI_label-WM_dseg.nii.gz'
>>> make_dipy_seeds.run()
```

**WM\_file** [a pathlike object or string representing a file] WM mask file registered to diffusion space.

**ROI\_files** [a list of items which are a pathlike object or string representing an existing file] ROI files registered to diffusion space.

**seed\_files** [a list of items which are a pathlike object or string representing an existing file] Seed files for probabilistic tractography.

```
Make_Seeds.ROI_idx = []
```

```
Make_Seeds.base_name = ''
```

```
Make_Seeds.gen_outputfilelist()
```

## SplitDiffusion

[Link to code](#)

Bases: `nipype.interfaces.base.core.BaseInterface`

Extract volumes from diffusion MRI data given a start and end index.

### Examples

```
>>> from cmtklib.diffusion import SplitDiffusion
>>> split_dwi = SplitDiffusion()
>>> split_dwi.inputs.in_file = 'sub-01_dwi.nii.gz'
>>> split_dwi.inputs.start = 5
>>> split_dwi.inputs.in_file = 30
>>> split_dwi.run()
```

**end** [an integer] Volume index to end the split.

**in\_file** [a pathlike object or string representing an existing file] Input diffusion MRI file.

**start** [an integer] Volume index to start the split.

**data** [a pathlike object or string representing an existing file] Extracted volumes.

**padding1** [a pathlike object or string representing a file] Extracted volumes with padding 1.

**padding2** [a pathlike object or string representing a file] Extracted volumes with padding 2.

## 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: `nipy.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, savefilename='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** `leng` – 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 filename or object implementing the os.PathLike interface] FMRI volume to detrend.

**gm\_file** [a list of items which are a filename or object implementing the os.PathLike interface] ROI files registered to fMRI space.

**mode** ['linear' or 'quadratic' or 'cubic'] Detrending order.

**out\_file** [a filename or object implementing the os.PathLike interface] Detrended fMRI volume.



## Discard\_tp

[Link to code](#)

Bases: `nipype.interfaces.base.core.BaseInterface`

Discards the `n` first time frame in functional MRI data.

### Examples

```
>>> from cmtklib.functionalMRI import Discard_tp
>>> discard = Discard_tp()
>>> 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 filename or object implementing the `os.PathLike` interface] Input 4D fMRI image.

**n\_discard** [an integer] Number of `n` first frames to discard.

**out\_file** [a filename or object implementing the `os.PathLike` interface] Output 4D fMRI image with discarded frames.

## Nuisance\_regression

[Link to code](#)

Bases: `nipype.interfaces.base.core.BaseInterface`

Regress out nuisance signals (WM, CSF, movements) through GLM.

### Examples

```
>>> from cmtklib.functionalMRI import Nuisance_regression
>>> nuisance = Nuisance_regression()
>>> 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',
>>>                               '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↳desc-scale5_dseg.nii.gz']
```

(continues on next page)

(continued from previous page)

```
>>> 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 filename or object implementing the `os.PathLike` interface] Eroded brain mask registered to fMRI space.

**csf\_file** [a filename or object implementing the `os.PathLike` interface] 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 filename or object implementing the `os.PathLike` interface] GM atlas files registered to fMRI space.

**in\_file** [a filename or object implementing the `os.PathLike` interface] Input fMRI volume.

**motion\_file** [a filename or object implementing the `os.PathLike` interface] 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 filename or object implementing the `os.PathLike` interface] Eroded WM mask registered to fMRI space.

**wm\_nuisance** [a boolean] If `True` perform WM nuisance regression.

**averageCSF\_mat** [a filename or object implementing the `os.PathLike` interface] Output matrix of CSF regression.

**averageCSF\_npy** [a filename or object implementing the `os.PathLike` interface] Output of CSF regression in `numpy` format.

**averageGlobal\_mat** [a filename or object implementing the `os.PathLike` interface] Output matrix of global regression.

**averageGlobal\_npy** [a filename or object implementing the `os.PathLike` interface] Output of global regression in `numpy` format.

**averageWM\_mat** [a filename or object implementing the `os.PathLike` interface] Output matrix of WM regression.

**averageWM\_npy** [a filename or object implementing the `os.PathLike` interface] Output of WM regression in `numpy` format.

**out\_file** [a filename or object implementing the `os.PathLike` interface] 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 filename or object implementing the `os.PathLike` interface] FMRI volume to scrub.

**gm\_file** [a list of items which are a filename or object implementing the `os.PathLike` interface] ROI volumes registered to fMRI space.

**motion\_parameters** [a filename or object implementing the `os.PathLike` interface] Motion parameters from preprocessing stage.

**wm\_mask** [a filename or object implementing the `os.PathLike` interface] WM mask registered to fMRI space.

**dvars\_mat** [a filename or object implementing the `os.PathLike` interface] DVARS matrix for scrubbing.

**dvars\_npy** [a filename or object implementing the `os.PathLike` interface] DVARS in `.npy` format.

**fd\_mat** [a filename or object implementing the `os.PathLike` interface] FD matrix for scrubbing.

**fd\_npy** [a filename or object implementing the `os.PathLike` interface] 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] Parcel-lation 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: `nipy.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 'Lausanne2008' or 'Lausanne2018'] 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.

## Erode

[Link to code](#)

Bases: nipype.interfaces.base.core.BaseInterface

Erodes a mask.

## Examples

```
>>> erode = Erode()
>>> erode.inputs.in_file = '/path/to/sub-01_desc-brain_mask.nii.gz'
>>> erode.run()
```

**in\_file** [a pathlike object or string representing an existing file] Input mask to erode.

**out\_file** [a pathlike object or string representing an existing file] Eroded mask.

## 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 *cmrtoolkit.parcellation* module for all parcellation resolutions of a given scheme.

## Example

```
>>> from cmrtoolkit.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** ['Lausanne2008' or 'Lausanne2018' or 'NativeFreesurfer'] Parcellation scheme. (Nipype **default** value: Lausanne2008)

**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: `nipy.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: `nipy.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\_hippocsubfields** [a pathlike object or string representing a file] Left hemisphere hippocampal subfields file.

**rh\_hippocsubfields** [a pathlike object or string representing a file] Right hemisphere hippocampal subfields file.

## ParcellateThalamus

[Link to code](#)

Bases: `nipy.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).

`cmcklib.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)

`cmcklib.parcellation.create_annot_label(subject_id, subjects_dir)`

Creates annotation labels for the Lausanne2008 parcellation scheme.

#### Parameters

- **subject\_id** (*string*) – Freesurfer subject id
- **subjects\_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output\_dir/freesurfer)

`cmcklib.parcellation.create_roi(subject_id, subjects_dir)`

Iteratively creates the ROI\_%s.nii.gz files using the given Lausanne2008 parcellation information from networks.

#### Parameters

- **subject\_id** (*string*) – Freesurfer subject id
- **subjects\_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output\_dir/freesurfer)

`cmcklib.parcellation.create_roi_v2(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)`

Creates the white-matter mask using the Freesurfer ribbon as basis in the Lausanne2008 framework.

#### Parameters

- **subject\_id** (*string*) – Freesurfer subject id
- **subjects\_dir** (*string*) – Freesurfer subjects dir (Typically `/path/to/output_dir/freesurfer`)

`cmtklib.parcellation.create_wm_mask_v2(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(parcelation_scheme, subject_id, subjects_dir)`

Convert Freesurfer images back to original native space when Lausanne20XX parcellation schemes are used.

#### Parameters

- **parcelation\_scheme** (*string*) – Parcellation scheme: ‘Lausanne2008’, ‘Lausanne2018’
- **subject\_id** (*string*) – Freesurfer subject id
- **subjects\_dir** (*string*) – Freesurfer subjects dir (Typically `/path/to/output_dir/freesurfer`)

`cmtklib.parcellation.define_atlas_variables()`

Returns a dictionary containing atlas information for the Lausanne2018 parcellation scheme.

---

**Note:** `atlas_info` often used in the code refers to such a dictionary.

---

`cmtklib.parcellation.erode_mask(fsd_dir, maskFile)`

Erodes the mask and saves it the Freesurfer subject directory.

#### Parameters

- **fsd\_dir** (*string*) – Freesurfer subject directory

- **maskFile** (*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.generate_single_parcellation` (*v, i, fs\_string, subject\_dir, subject\_id*)  
Generates the volumetric parcellation from the annotation file for one scale of Lausanne2018 parcellation.

**Parameters**

- **v** (*Boolean*) – Verbose mode
- **i** (*int*) – Parcellation scale index
- **fs\_string** (*string*) – Command executed before running Freesurfer tool
- **subject\_id** (*string*) – Freesurfer subject id
- **subject\_dir** (*string*) – Freesurfer subject dir (Typically `/path/to/output_dir/freesurfer/<subject_id>`)

`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', 'Lausanne2008' 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.bidsapp\_2\_local\_bids\_dir(*local\_dir*, *path*, *debug=True*)

Replace all path prefixes /bids\_dir by the real local directory path.

### Parameters

- **local\_dir** (*string*) – Local path to BIDS root directory
- **path** (*string*) – Path where the prefix should be replaced
- **debug** (*Boolean*) – If True, print the new path after replacement

**Returns** *new\_path* – Output path with prefix replaced

**Return type** *string*

cmtklib.util.bidsapp\_2\_local\_output\_dir2(*local\_dir*, *path*, *debug=True*)

Replace all path prefixes /output\_dir by the real local directory path.

### Parameters

- **local\_dir** (*string*) – Local path to output / derivatives directory
- **path** (*string*) – Path where the prefix should be replaced
- **debug** (*Boolean*) – If True, print the new path after replacement

**Returns** *new\_path* – Output path with prefix replaced

**Return type** *string*

cmtklib.util.create\_results\_plkz\_local(*plkz\_file*, *local\_output\_dir*, *debug=True*)

Update path in pickle files generated by Nipype nodes.

### Parameters

- **plkz\_file** (*zipped pickle*) – Pickle file generated by Nipype node
- **local\_output\_dir** (*string*) – Local output / derivatives directory
- **debug** (*Boolean*) – If True, print extra information

cmtklib.util.extract\_freesurfer\_subject\_dir(*reconall\_report*, *local\_output\_dir=None*)

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

**Returns** **fs\_subject\_dir** – Freesurfer subject directory

**Return type** *string*

`cmtklib.util.force_decode` (*string*, *codecs=None*)

Force decoding byte string with a specific codec.

**Parameters**

- **string** (*bytes*) – String in bytearray format to be decoded
- **codecs** (*list*) – List of codecs to try to decode the encoded string

`cmtklib.util.get_node_dictionary_outputs` (*node\_report*, *local\_output\_dir=None*)

Read the Nipype node report and return a dictionary of node outputs.

**Parameters**

- **node\_report** (*string*) – Path to node report
- **local\_output\_dir** (*string*) – Local output / derivatives directory

**Returns** **dict\_outputs** – dictionary of outputs extracted from node execution report

**Return type** *dict*

`cmtklib.util.get_pipeline_dictionary_outputs` (*datasink\_report*, *local\_output\_dir=None*)

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

**Returns** **dict\_outputs** – Dictionary of pipeline outputs

**Return type** *dict*

`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.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** (*list*) – Parcellation scheme  
(`['NativeFreesurfer', 'Lausanne2008', 'Lausanne2018']`)
- **weight** (*list*) – Edge metric to extract from the graph  
(`['number_of_fibers', 'fiber_density', ...]`)

**Returns** `connmats` – Dictionary of connectivity matrices

**Return type** `dict`

`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.return_button_style_sheet` (*image*, *icon\_size*, *verbose=False*)

Return Qt style sheet for QPushButton with image

### Parameters

- **image** (*string*) – Path to image to use as icon
- **icon\_size** (*int*) – Image size
- **verbose** (*Bool*) – Print the style sheet if True Default: False

**Returns** `button_style_sheet` – Qt style sheet for QPushButton with image

**Return type** string

## 5.7 Adopting Datalad for collaboration

Datalad is a powerful tool for the versioning and sharing 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 the use of datalad datasets with the connectome mapper in in our lab at the time of creation of this document (2019 Jan 8). For more details and tutorials on Datalad, please check the recent [Datalad Handbook](#)

**Warning:** This was tested on Ubuntu 16.04 with Datalad 0.11.3 and its extensions datalad-container 0.3.1, datalad\_neuroimaging 0.2.0 and datalad\_revolution 0.6.0. This example might not work with their latest versions as they are under intensive development and a number of new versions with minor and major changes have been released in the meantime.

### 5.7.1 Move original BIDS dataset to server

```
rsync -P -v -avz -e 'ssh' --exclude 'derivatives' --exclude 'code' --exclude '.datalad  
→ ' --exclude '.git' --exclude '.gitattributes' /media/localadmin/HagmannHDD/Seb/ds-  
→ newtest2/* tourbier@<SERVER_IP_ADDRESS>:/home/tourbier/Data/ds-newtest2
```

### 5.7.2 Datalad setup and dataset creation on Server (accessible via ssh)

#### Connect to server

```
ssh tourbier@<SERVER_IP_ADDRESS>
```

#### Install git-annex and liblzma-dev (datalad dependencies), Datalad and its extensions

```
sudo apt-get install git-annex liblzma-dev  
pip install datalad[all]==0.11.3  
pip install datalad-container==0.3.1  
pip install datalad_neuroimaging==0.2.0  
pip install datalad_revolution==0.6.0
```

---

**Note:** Tested using git-annex version 7.20190219-gad7c11b

---

#### Go to source dataset directory, create a Datalad dataset and save all

```
cd /home/tourbier/Data/ds-newtest2  
datalad rev-create -f -D "Original test dataset on lab server"  
datalad rev-save -m 'Source (Origin) BIDS dataset' --version-tag origin
```

#### Report on the state of dataset content

```
datalad rev-status --recursive
```

### 5.7.3 Processing using the Connectome Mapper BIDS App on a local workstation

#### Dataset installation

```
datalad install -s ssh://tourbier@<SERVER_IP_ADDRESS>:/home/tourbier/Data/ds-newtest2_  
→ \  
/home/localadmin/Data/ds-newtest2  
  
cd /home/localadmin/Data/ds-newtest2
```



## Get T1w and Diffusion images to be processed, written in a bash script for reproducibility

```
datalad get -J 4 sub-*/ses-*/anat/sub-*_T1w.nii.gz
datalad get -J 4 sub-*/ses-*/dwi/sub-*_dwi.nii.gz
datalad get -J 4 sub-*/ses-*/dwi/sub-*_dwi.bvec
datalad get -J 4 sub-*/ses-*/dwi/sub-*_dwi.bval
```

Write datalad get commands to `get_required_files_for_analysis.sh`:

```
mkdir code
echo "datalad get -J 4 sub-*/ses-*/anat/sub-*_T1w.nii.gz" > code/get_required_files_
↪for_analysis.sh
echo "datalad get -J 4 sub-*/ses-*/dwi/sub-*_dwi.nii.gz" >> code/get_required_files_
↪for_analysis.sh
echo "datalad get -J 4 sub-*/ses-*/dwi/sub-*_dwi.bvec" >> code/get_required_files_for_
↪analysis.sh
echo "datalad get -J 4 sub-*/ses-*/dwi/sub-*_dwi.bval" >> code/get_required_files_for_
↪analysis.sh
```

Add all content in the `code/` directory directly to git:

```
datalad add --to-git code
```

## Add the container image of the connectome mapper to the dataset

```
datalad containers-add connectomemapper-bidsapp-|release| \
--url dhub://sebastientourbier/connectomemapper-bidsapp:|release| \
--update
```

## Save the state of the dataset prior to analysis

```
datalad rev-save -m "Seb's test dataset on local \
workstation ready for analysis with connectomemapper-bidsapp:|release|" \
--version-tag ready4analysis-<date>-<time>
```

## Run Connectome Mapper on all subjects

```
datalad containers-run --container-name connectomemapper-bidsapp-|release| \
'/tmp' '/tmp/derivatives' participant \
--anat_pipeline_config '/tmp/code/ref_anatomical_config.ini' \
--dwi_pipeline_config '/tmp/code/ref_diffusion_config.ini' \
```

## Save the state

```
datalad rev-save -m "Seb's test dataset on local \
workstation processed by connectomemapper-bidsapp:|release|, {Date/Time}" \
--version-tag processed-<date>-<time>
```

## Report on the state of dataset content

```
datalad rev-status --recursive
```

**With DataLad with don't have to keep those inputs around – without losing the ability to reproduce an analysis.**

Let's uninstall them – checking the size on disk before and after:

```
datalad uninstall sub-*/*
```

## 5.7.4 Local collaboration with Bob for Electrical Source Imaging

### Processed dataset installation on Bob's workstation

```
datalad install -s (ssh://)localadmin@HOS51827:/home/localadmin/Data/ds-newtest2 \
/home/bob/Data/ds-newtest2

cd /home/bob/Data/ds-newtest2
```

### Get connectome mapper output files (Brain Segmentation and Multi-scale Parcellation) used by Bob in his analysis

```
datalad get -J 4 derivatives/cmp/sub-*/ses-*/anat/sub-*_mask.nii.gz
datalad get -J 4 derivatives/cmp/sub-*/ses-*/anat/sub-*_class-*_dseg.nii.gz
datalad get -J 4 derivatives/cmp/sub-*/ses-*/anat/sub-*_scale*_atlas.nii.gz
```

Write datalad get commands to get\_required\_files\_for\_analysis\_by\_bob.sh for reproducibility:

```
echo "datalad get -J 4 derivatives/cmp/sub-*/ses-*/anat/sub-*_mask.nii.gz" > code/get_
↪required_files_for_analysis_by_bob.sh
echo "datalad get -J 4 derivatives/cmp/sub-*/ses-*/anat/sub-*_class-*_dseg.nii.gz" >>_
↪code/get_required_files_for_analysis_by_bob.sh
echo "datalad get -J 4 derivatives/cmp/sub-*/ses-*/anat/sub-*_scale*_atlas.nii.gz" >>_
↪code/get_required_files_for_analysis_by_bob.sh
```

Add all content in the code/ directory directly to git:

```
datalad add --to-git code
```

## Update derivatives

```
cd /home/bob/Data/ds-newtest2
mkdir derivatives/cartool ...
```

## Save the state

```
datalad rev-save -m "Bob's test dataset on local \
workstation processed by cartool:|release|, {Date/Time}" \
--version-tag processed-<date>-<time>
```

## Report on the state of dataset content

```
datalad rev-status --recursive
```

**With DataLad with don't have to keep those inputs around – without losing the ability to reproduce an analysis.**

Let's uninstall them – checking the size on disk before and after:

```
datalad uninstall sub-*/*
datalad uninstall derivatives/cmp/*
datalad uninstall derivatives/freesurfer/*
datalad uninstall derivatives/nipype/*
```

- Created by Sebastien Tourbier - 2019 Jan 8

## 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 3.5.1 and deployed to [Sylabs.io](https://sylabs.io) as (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:
➔v3.0.0-RC2
```

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](#).

### 5.8.1 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:v3.0.0-RC2_
→ \
  /bids_dir /output_dir participant --participant_label 01 02 03 \
  --anat_pipeline_config /bids_dir/code/ref_anatomical_config.ini \
  --dwi_pipeline_config /bids_dir/code/ref_diffusion_config.ini \
  --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.

---

### 5.8.2 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-v3.0.0-RC2.simg sebastientourbier/
→ connectomemapper-bidsapp:v3.0.0-RC2
```

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-v3.0.0-RC2.simg <your_cluster_user_login>@
→ <cluster_url>:~/Softwares/singularity/cmp-v3.0.0-RC2.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-v3.0.0-RC2.simg docker://
↪sebastientourbier/connectomemapper-bidsapp:v3.0.0-RC2
```

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 uncompatibilities

## 5.8.3 Useful singularity commands

- Display a container's metadata:

```
$ singularity inspect ~/Softwares/singularity/cmp-v3.0.0-RC2.simg
```

- Clean cache:

```
$ singularity cache clean
```

Created by Sebastien Tourbier - 2020 Mar 04 - Latest update: 2020 Dec 24

## 5.9 BSD 3-Clause License

Copyright (C) 2009-2020, Ecole Polytechnique Fédérale de Lausanne (EPFL) and Hospital Center and University of Lausanne (UNIL-CHUV), Switzerland, & Contributors, All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the Ecole Polytechnique Fédérale de Lausanne (EPFL) and Hospital Center and University of Lausanne (UNIL-CHUV) nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL "Ecole Polytechnique Fédérale de Lausanne (EPFL) and Hospital Center and University of Lausanne (UNIL-CHUV)" BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

**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.

## 5.10 Changes

### 5.10.1 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 occurring.
- 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.

## 5.10.2 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.

```
- nipy 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.

### 5.10.3 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.

### 5.10.4 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 meaningful 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.

### **5.10.5 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 mrtix3 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.

### **5.10.6 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)

### **5.10.7 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

## 5.11 Citing

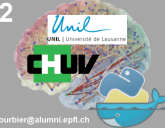
### Important:

- If you are using the Connectome Mapper 3 in your work, a manuscript is in preparation, but for now, please acknowledge this software with the following two entries:
1. Tourbier S, Aleman-Gomez Y, Mullier E, Griffa A, Bach Cuadra M, Hagmann P (2020). connectomic-slab/connectomemapper3: Connectome Mapper v3.0.0-RC2 (Version v3.0.0-RC2). Zenodo. <http://doi.org/10.5281/zenodo.3475969>.
  2. Tourbier S, Aleman-Gomez Y, Mullier E, Griffa A, Bach Cuadra M, Hagmann P (2020, June). Connectome Mapper 3: a software pipeline for multi-scale connectome mapping of multimodal MR data. 26th Annual Meeting of the Organization for Human Brain Mapping (OHBM), abstract #1174, poster #1892.

### 5.11.1 Poster

- Organization for Human Brain Mapping 2020 (Abstract; Poster)

# 1892



Connectome Mapper 3: a software pipeline for multi-scale connectome mapping of multimodal MR data

S. Tourbier<sup>1</sup>, Y. Alemán-Gómez<sup>1,4</sup>, E. Mullier<sup>1</sup>, A. Griffa<sup>2</sup>, M. Bach Cuadra<sup>1,3</sup>, P. Hagmann<sup>1</sup>

1. Department of Radiology, University Hospital of Lausanne (CHUV) and University of Lausanne (UNIL), Lausanne, Switzerland  
2. Medical Image Processing Lab (MIP/LAB), Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland  
3. Medical Image Analysis Laboratory (MIAL), Centre d'Imagerie BioMédicale (CIBM), Lausanne, Switzerland  
4. Institut de Neurosciences de lausanne (INSL)

sebastien.tourbier@chuv.ch

## MOTIVATION

Connectome Mapper (CMP) [1,2]: open-source software pipeline, written in Python with a Graphical User Interface (GUI) historically designed to help researchers in all the organization and processing steps needed to compute, from raw structural MRI (sMRI) and diffusion MRI (dMRI) data, a hierarchical multi-scale brain parcellation and the corresponding structural connectomes.

Designed with ease-of-use, modularity, configurability, re-executability and transparency in mind.

But two previous versions limited in terms of interoperability and reusability, portability, and reproducibility, with dependencies not easy to install.

## OBJECTIFS

- Provide a unique software pipeline solution with a GUI for researchers to easily, reliably and transparently create a hierarchical multi-scale connectome representation of the structural and functional brain systems.
- Adopt recent advances in the standardization of neuroimaging data organization [2] and processing [4-6] that:
  - promote processing interoperability, reusability, portability and reproducibility, and
  - have proven to be capable of effective large scale collaboration

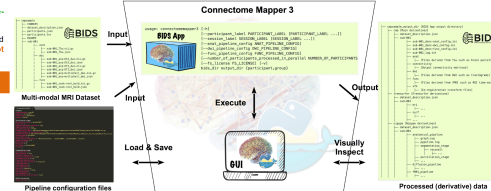
## HIGHLIGHTS

- Extend the 5-scale brain gray matter parcellation [2] derived from the Desikan-Killiany atlas [8] with:
  - a subdivision of the thalamus into 7 nuclei [7],
  - the hippocampus into 12 subfields [9] and
  - the brainstem into 4 sub-structures [6].
- Read datasets following the Brain Imaging Data Structure (BIDS) standard [3].
- Outputs (derivatives) structured according to BIDS derivatives draft (see complete list)
  - interoperable, reusable
- Use ConfigParser, Traits and Nipype [4] to implement a modular processing workflow composed of configurable pipelines, each dedicated to one modality (anatomical, diffusion, fMRI), that interface with tools from FSL[10], FreeSurfer[11], ANTs[12], Dipy[13], MRtrix3[14], AFNI[15], and the in-house Connectome Mapper library (cmxlib).
- modular, configurable, scalable, re-executable with data provenance tracking
- Workflow encapsulated with all dependencies in a software container image following the BIDS Apps standard [5].
  - easy-to-be-installed, easy-to-interface, portable, scalable and reproducible
- Provide a GUI to ease and support all the steps involved in (1) the configuration of the pipeline, (2) the configuration of the BIDS App run and its execution, and (3) the inspection of the stage outputs.
- Distributed under the 3-Clause BSD license
- Hosted on GitHub @ <https://github.com/connectomeconnectomeconnectomemapper3> where issues and new feature requests are transparently discussed, and versions are released through continuous integration testing
  - modern community-driven software practices

## ARCHITECTURE

Software bundle which contains:

- The Connectome Mapper BIDS App: the processing core encapsulating the processing workflow in a software container image (Docker [16] and Singularity [17]) that takes as main input a BIDS dataset and specific pipeline configuration files.
- The Connectome Mapper BIDS App Manager: the user-friendly graphical user interface (GUI) which relies on the traitlets/Jupyter windowing framework, and allows you to easily configure and run the pipelines and inspect the outputs.



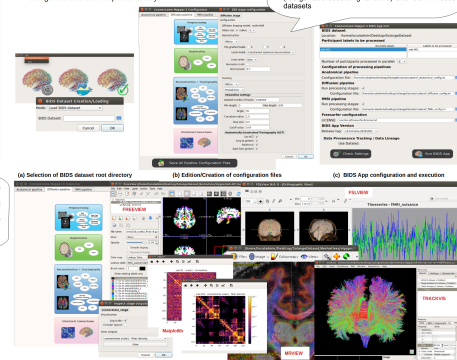
## GRAPHICAL USER INTERFACE

A typical use case consists only of a few steps which are:

- (a) the selection of the root directory of the BIDS dataset to be analyzed,
- (b) the creation/editing of each pipeline configuration file,
- (c) the configuration of the BIDS App run and its execution,
- (d) the inspection of the stage outputs using either follow-up, or review, or miniview or trackvis depending on the software involved in the stage. Connectivity matrices are visualized using nilearn and the matplotlib library.

While the configuration files can be easily created from the GUI, the execution of the BIDS App can also be scripted to guarantee a consistent and homogeneous processing on a collection of datasets at the same time.

Adopting this procedure, CMP3 has been successfully run on the range of supported diffusion schemes using the HCP test-retest (reorganized according to BIDS) and four in-house datasets.

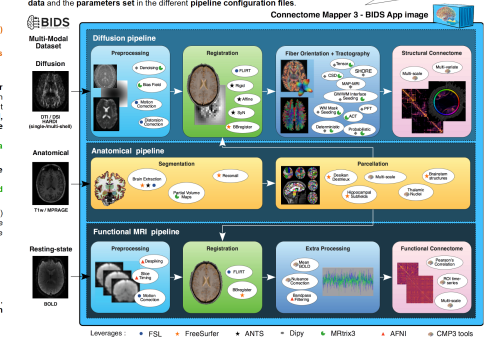


## PROCESSING WORKFLOW

Thanks to its modular architecture and the use of Nipype and tractsch/tractools, new pipelines and stages with GUI components can be added with relatively little effort to account for additional imaging modalities and algorithms.

Take any BIDS datasets as input that include at least a structural MRI (sMRI) scan (T1w or T2w) and a diffusion MRI (dMRI) scan (DTI, CSD, single/multi-shell HARDI) and/or a resting-state fMRI (rsfMRI) scan.

Dynamically built and configured depending on the availability of sMRI, dMRI, fMRI data and the parameters set in the different pipeline configuration files.

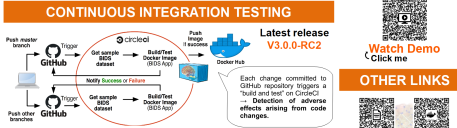


Leverages: FSL, FreeSurfer, ANTs, Dipy, MRtrix3, AFNI, CMP3 tools

## CONTINUOUS INTEGRATION TESTING

Latest release V3.0.0-RC2

Each change committed to GitHub repository triggers a "build and test" on GitHub Actions. Detection of adverse effects arising from code changes.



## UPCOMING

- Implementation of EEG pipeline initiated during Global Brainhack Geneva 2019 (see Project) and discussed in the following GitHub issue.
- BIDS and CMP3 Training at the upcoming ReproKIMinimizing Summer School 2020 that will be held in Lausanne, Switzerland (expected to June 2021, date to be confirmed). See website for more details.

## OTHER LINKS

License BIDS App

Code Docs

For latest news and support, please also consult our CMTX User Group forum.

[1] Daducci et al., PLoS ONE 2013. [2] Cammoun et al., J. Neuro. Methods 2012. [3] Gorgolewski et al., Scientific Data 2016. [4] Gorgolewski et al., Front. Neuro. 2011. [5] Gorgolewski et al., PLoS CB 2017. [6] Desikan et al., Neuroimage 2004. [7] Napadow et al., Scientific Data 2018. [8] Desikan et al., Neuroimage 2010. [9] Jernigan et al., Neuroimage 2012. [10] Jenkinson et al., Neuroimage 2012. [11] Fischl et al., Neuroimage 2012. [12] Avants et al., IMA 2016. [13] Dipy et al., Front. Neuroinform 2014. [14] Tourbier et al., Neuroimage 2019. [15] Cox et al., Computers and Biomedical Research 1996. [16] Menet et al., Linux. Journal 2014. [17] Rattier et al., PLoS ONE 2017.

## 5.12 Contributing to Connectome Mapper 3

### Contents

- *Contributing to Connectome Mapper 3*
  - *Philosophy*
  - *Contribution Types*
    - \* *Report Bugs*
    - \* *Fix Bugs*
    - \* *Implement Features*
    - \* *Write Documentation*
    - \* *Submit Feedback*
  - *Get Started!*
    - \* *Pull Request Guidelines*
    - \* *How to build the BIDS App locally*
    - \* *How to build the documentation locally*

### 5.12.1 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 Connectome 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.
- 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 CMP 3 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.

## 5.12.2 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.

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.

### 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.

## 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 :)

## 5.12.3 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 unstaged 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.

---

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 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

1. Install the CMP3 conda environment `py37cmp-gui` with `sphinx` and all extensions to generate the documentation:

```
cd connectomemapper3
conda env create -f environment.yml
```

2. Activate CMP3 conda environment `py37cmp-gui` and install `connectomemapper3`

```
conda activate py37cmp-gui
(py37cmp-gui) python setup.py install
```

3. Run the script `build_docs.sh` to generate the HTML documentation in `docs/_build/html`:

```
(py37cmp-gui) bash build_docs.sh
```

---

**Note:** Make sure to have activated the conda environment `py37cmp-gui` before running the script `build_docs.sh`.

---

**Authors** Sebastien Tourbier, Adrien Birbaumer

**Version** Revision: 2

**Copyright** Copyright (C) 2017-2020, Ecole Polytechnique Federale de Lausanne (EPFL) the University Hospital Center and University of Lausanne (UNIL-CHUV), Switzerland, & Contributors All rights reserved..

### Acknowledgments

We thanks the authors of [these great contributing guidelines](#), from which part of this document has been inspired and adapted.

## 5.13 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)



## FUNDING

Work supported by the SNF Sinergia Grant 170873 (<http://p3.snf.ch/Project-170873>).



## BIBLIOGRAPHY

- [Tournier2007] Tournier, J.D., et al. NeuroImage 2007. Robust determination of the fibre orientation distribution in diffusion MRI: Non-negativity constrained super-resolved spherical deconvolution
- [Merlet2013] Merlet S. et. al, Medical Image Analysis, 2013.
- [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](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

### C

cmp, 79  
 cmp.bidsappmanager.core, 116  
 cmp.bidsappmanager.gui, 117  
 cmp.bidsappmanager.pipelines.anatomical, 130  
 cmp.bidsappmanager.pipelines.anatomical.anatomical, 130  
 cmp.bidsappmanager.pipelines.diffusion, 131  
 cmp.bidsappmanager.pipelines.diffusion.diffusion, 131  
 cmp.bidsappmanager.pipelines.functional, 132  
 cmp.bidsappmanager.pipelines.functional.eeg, 132  
 cmp.bidsappmanager.pipelines.functional.fMRI, 133  
 cmp.bidsappmanager.project, 125  
 cmp.bidsappmanager.stages, 134  
 cmp.bidsappmanager.stages.connectome, 134  
 cmp.bidsappmanager.stages.connectome.connectome, 134  
 cmp.bidsappmanager.stages.connectome.fmri\_connectome, 135  
 cmp.bidsappmanager.stages.diffusion, 136  
 cmp.bidsappmanager.stages.diffusion.diffusion, 136  
 cmp.bidsappmanager.stages.diffusion.reconstruction, 137  
 cmp.bidsappmanager.stages.diffusion.tracking, 137  
 cmp.bidsappmanager.stages.functional, 138  
 cmp.bidsappmanager.stages.functional.functionalMRI, 138  
 cmp.bidsappmanager.stages.parcellation, 139  
 cmp.bidsappmanager.stages.parcellation.parcellation, 139  
 cmp.bidsappmanager.stages.preprocessing, 139  
 cmp.bidsappmanager.stages.preprocessing.fmri\_preprocessing, 139  
 cmp.bidsappmanager.stages.preprocessing.preprocess, 140  
 cmp.bidsappmanager.stages.registration, 141  
 cmp.bidsappmanager.stages.registration.registration, 141  
 cmp.bidsappmanager.stages.segmentation, 142  
 cmp.bidsappmanager.stages.segmentation.segmentation, 142  
 cmp.parser, 79  
 cmp.pipelines.anatomical, 86  
 cmp.pipelines.anatomical.anatomical, 86  
 cmp.pipelines.common, 85  
 cmp.pipelines.diffusion, 88  
 cmp.pipelines.diffusion.diffusion, 88  
 cmp.pipelines.functional, 90  
 cmp.pipelines.functional.eeg, 90  
 cmp.pipelines.functional.fMRI, 90  
 cmp.project, 80  
 cmp.stages, 92  
 cmp.stages.common, 115  
 cmp.stages.connectome, 92  
 cmp.stages.connectome.connectome, 92  
 cmp.stages.connectome.fmri\_connectome, 93  
 cmp.stages.diffusion, 95  
 cmp.stages.diffusion.diffusion, 95  
 cmp.stages.diffusion.reconstruction, 98  
 cmp.stages.diffusion.tracking, 100  
 cmp.stages.functional, 103  
 cmp.stages.functional.functionalMRI, 103  
 cmp.stages.parcellation, 104  
 cmp.stages.parcellation.parcellation, 104  
 cmp.stages.preprocessing, 106  
 cmp.stages.preprocessing.fmri\_preprocessing, 106  
 cmp.stages.preprocessing.preprocessing, 106

107  
cmp.stages.registration, 109  
cmp.stages.registration.registration,  
109  
cmp.stages.segmentation, 113  
cmp.stages.segmentation.segmentation,  
113  
cmtklib, 143  
cmtklib.bids, 143  
cmtklib.bids.utils, 143  
cmtklib.config, 195  
cmtklib.connectome, 199  
cmtklib.diffusion, 202  
cmtklib.functionalMRI, 208  
cmtklib.interfaces, 143  
cmtklib.interfaces.afni, 143  
cmtklib.interfaces.ants, 146  
cmtklib.interfaces.camino, 147  
cmtklib.interfaces.camino2trackvis, 150  
cmtklib.interfaces.diffusion\_toolkit,  
152  
cmtklib.interfaces.dipy, 155  
cmtklib.interfaces.freesurfer, 161  
cmtklib.interfaces.fsl, 165  
cmtklib.interfaces.misc, 174  
cmtklib.interfaces.mrtrix3, 175  
cmtklib.parcellation, 212  
cmtklib.util, 220

## A

aggregate\_outputs() (cmklib.interfaces.fsl.Orient method), 174  
 anat\_available\_config (cmp.project.CMP\_Project\_Info attribute), 81  
 anat\_config (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow attribute), 118  
 anat\_config\_error\_msg (cmp.project.CMP\_Project\_Info attribute), 81  
 anat\_config\_error\_view (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 124  
 anat\_config\_to\_load (cmp.project.CMP\_Project\_Info attribute), 81  
 anat\_custom\_last\_stage (cmp.project.CMP\_Project\_Info attribute), 82  
 anat\_flow (cmp.pipelines.common.Pipeline attribute), 85  
 anat\_inputs\_checked (cmp.bidsappmanager.gui.CMP\_ConfiguratorWindow attribute), 120  
 anat\_inputs\_checked (cmp.bidsappmanager.gui.CMP\_InspectorWindow attribute), 121  
 anat\_inputs\_checked (cmp.bidsappmanager.project.CMP\_ConfigQualityWindow attribute), 126  
 anat\_inputs\_checked (cmp.bidsappmanager.project.CMP\_MainWindowHandler attribute), 128  
 anat\_load\_config\_ini() (in module cmklib.config), 195  
 anat\_load\_config\_json() (in module cmklib.config), 195  
 anat\_outputs\_checked (cmp.bidsappmanager.project.CMP\_ConfigQualityWindow attribute), 126  
 anat\_outputs\_checked (cmp.bidsappmanager.project.CMP\_MainWindowHandler attribute), 128  
 anat\_pipeline (cmp.bidsappmanager.gui.CMP\_ConfiguratorWindow attribute), 120  
 anat\_pipeline (cmp.bidsappmanager.gui.CMP\_InspectorWindow attribute), 121  
 anat\_pipeline (cmp.bidsappmanager.gui.CMP\_MainWindow attribute), 122  
 anat\_pipeline (cmp.bidsappmanager.project.CMP\_ConfigQualityWindow attribute), 126  
 anat\_pipeline (cmp.bidsappmanager.project.CMP\_MainWindowHandler attribute), 128  
 anat\_run (cmp.project.CMP\_Project\_Info attribute), 80  
 anat\_runs (cmp.project.CMP\_Project\_Info attribute), 80  
 anat\_save\_config (cmp.bidsappmanager.gui.CMP\_ConfiguratorWindow attribute), 120  
 anat\_save\_config() (in module cmklib.config), 196  
 anat\_select\_config\_to\_load (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 124  
 anat\_stage\_names (cmp.project.CMP\_Project\_Info attribute), 82  
 anat\_warning\_view (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 124  
 anatomical\_processed (cmp.bidsappmanager.project.CMP\_ConfigQualityWindow attribute), 126  
 anatomical\_processed (cmp.bidsappmanager.project.CMP\_MainWindowHandler attribute), 128  
 AnatomicalPipeline (class in cmp.pipelines.anatomical.anatomical), 86  
 AnatomicalPipelineUI (class in cmp.bidsappmanager.pipelines.anatomical.anatomical), 130  
 angle (cmp.stages.diffusion.tracking.MRtrix\_tracking\_config attribute), 102  
 ants\_bspline\_interpolation\_parameters

(cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 ants\_convergence\_thresh (cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 ants\_convergence\_winsize (cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 ants\_gauss\_interpolation\_parameters (cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 ants\_interpolation (cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 ants\_linear\_cost (cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 ants\_linear\_gradient\_step (cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 ants\_linear\_sampling\_perc (cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 ants\_linear\_sampling\_strategy (cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 ants\_lower\_quantile (cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 ants\_multilab\_interpolation\_parameters (cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 ants\_nonlinear\_cost (cmp.stages.registration.registration.RegistrationConfig attribute), 111  
 ants\_nonlinear\_gradient\_step (cmp.stages.registration.registration.RegistrationConfig attribute), 111  
 ants\_nonlinear\_total\_field\_variance (cmp.stages.registration.registration.RegistrationConfig attribute), 111  
 ants\_nonlinear\_update\_field\_variance (cmp.stages.registration.registration.RegistrationConfig attribute), 111  
 ants\_perform\_syn (cmp.stages.registration.registration.RegistrationConfig attribute), 111  
 ants\_precision\_type (cmp.stages.parcellation.parcellation.ParcellationConfig attribute), 105  
 ants\_probmaskfile (cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 113  
 ants\_regmaskfile (cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 114  
 ants\_templatefile (cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 113  
 ants\_upper\_quantile (cmp.stages.registration.registration.RegistrationConfig attribute), 110  
 apply\_scrubbing (cmp.stages.connectome.fmri\_connectome.ConnectomeConfig attribute), 93  
 apply\_to\_eroded\_brain (cmp.stages.registration.registration.RegistrationConfig attribute), 112  
 apply\_to\_eroded\_csf (cmp.stages.registration.registration.RegistrationConfig attribute), 112  
 apply\_to\_eroded\_wm (cmp.stages.registration.registration.RegistrationConfig attribute), 111  
 ApplymultipleMRConvert, 175  
 ApplymultipleMRCrop, 176  
 ApplymultipleMRTransforms, 176  
 ApplymultipleWarp, 165  
 ApplymultipleXfm, 165  
 atlas\_info (cmp.project.CMP\_Project\_Info attribute), 81  
 atlas\_info (cmp.stages.parcellation.parcellation.ParcellationConfig attribute), 105  
 B  
 backtrack (cmp.stages.diffusion.tracking.MRtrix\_tracking\_config attribute), 102  
 bidsapp, 143  
 base\_directory (cmp.project.CMP\_Project\_Info attribute), 80  
 base\_name (cmtklib.diffusion.Make\_Mrtrix\_Seeds attribute), 205  
 base\_name (cmtklib.diffusion.Make\_Seeds attribute), 205  
 BBRegister, 161  
 BColors (class in cmtklib.util), 220  
 bids\_field\_algo (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 108  
 bias\_field\_correction (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 108  
 bids (cmp.stages.common.Stage attribute), 115  
 bids\_layout (cmp.project.CMP\_Project\_Info attribute), 80  
 bids\_root (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow attribute), 117  
 bidsapp (cmp.bidsappmanager.gui.CMP\_MainWindow attribute), 122  
 bidsapp\_2\_local\_bids\_dir() (in module cmtklib.util), 220  
 bidsapp\_2\_local\_output\_dir2() (in module cmtklib.util), 220



**bidsapp\_tag**  (*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow* attribute), 118  
 **bidsapp\_ui**  (*cmp.bidsappmanager.gui.CMP\_MainWindow* attribute), 122  
 **big\_delta**  (*cmp.stages.diffusion.reconstruction.Dipy\_reconstruction* attribute), 99  
 **BinaryThreshold** , 166  
 **BOLD**  (*cmklib.util.BColors* attribute), 220  
 **brain\_mask\_extraction\_tool**  (*cmp.stages.segmentation.segmentation.SegmentationConfig* attribute), 113  
 **brain\_mask\_path**  (*cmp.stages.segmentation.segmentation.SegmentationConfig* attribute), 114  
 **C**   
 **Camino2Trackvis** , 150  
 **check**  (*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow* attribute), 118  
 **check\_config**  (*cmp.pipelines.anatomical.anatomical.AnatomicalPipeline* method), 86  
 **check\_config**  (*cmp.pipelines.common.Pipeline* method), 85  
 **check\_config**  (*cmp.pipelines.diffusion.diffusion.DiffusionPipeline* method), 88  
 **check\_config**  (*cmp.pipelines.functional.fMRI.fMRIPipeline* method), 91  
 **check\_input**  (*cmp.bidsappmanager.pipelines.anatomical.anatomical.AnatomicalPipelineUI* method), 131  
 **check\_input**  (*cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipelineUI* method), 132  
 **check\_input**  (*cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipelineUI* method), 133  
 **check\_input**  (*cmp.pipelines.anatomical.anatomical.AnatomicalPipeline* method), 86  
 **check\_input**  (*cmp.pipelines.diffusion.diffusion.DiffusionPipeline* method), 88  
 **check\_input**  (*cmp.pipelines.functional.fMRI.fMRIPipeline* method), 91  
 **Check\_Input\_Notification**  (class in *cmp.bidsappmanager.pipelines.functional.eeg* module), 87  
 **Check\_Input\_Notification**  (class in *cmp.bidsappmanager.pipelines.functional.fMRI* module), 88  
 **Check\_Input\_Notification**  (class in *cmp.bidsappmanager.project* module), 90  
 **Check\_Input\_NotificationUI**  (class in *cmp.bidsappmanager.pipelines.anatomical.anatomical* module), 131  
 **Check\_Input\_NotificationUI**  (class in *cmp.bidsappmanager.pipelines.diffusion.diffusion* module), 131  
 **Check\_Input\_NotificationUI**  (class in *cmp.bidsappmanager.pipelines.functional.fMRI* module), 133  
 **clean\_cache**  (in *cmp.bidsappmanager.project* module), 128  
 **clear\_stage\_outputs**  (*cmp.pipelines.common.Pipeline* method), 85  
 **cmat**  (in module *cmklib.connectome*), 200  
 **cmp**   
 **cmp**  module, 79  
 **cmp.bidsappmanager.core**  module, 116  
 **cmp.bidsappmanager.gui**  module, 117  
 **cmp.bidsappmanager.pipelines**  module, 130  
 **cmp.bidsappmanager.pipelines.anatomical**  module, 130  
 **cmp.bidsappmanager.pipelines.diffusion**  module, 131  
 **cmp.bidsappmanager.pipelines.diffusion.diffusion**  module, 131  
 **cmp.bidsappmanager.pipelines.functional**  module, 132  
 **cmp.bidsappmanager.pipelines.functional.eeg**  module, 132  
 **cmp.bidsappmanager.pipelines.functional.fMRI**  module, 133  
 **cmp.bidsappmanager.project**  module, 125  
 **cmp.bidsappmanager.stages**  module, 134  
 **cmp.bidsappmanager.stages.connectome**  module, 134  
 **cmp.bidsappmanager.stages.connectome.connectome**  module, 134  
 **cmp.bidsappmanager.stages.connectome.fmri\_connector**  module, 135  
 **cmp.bidsappmanager.stages.diffusion**  module, 136

<code>cmp.bidsappmanager.stages.diffusion.diffusion</code>	<code>cmp.stages.connectome.connectome</code>
module, 136	module, 92
<code>cmp.bidsappmanager.stages.diffusion.reconstruction</code>	<code>cmp.stages.connectome.fmri_connectome</code>
module, 137	module, 93
<code>cmp.bidsappmanager.stages.diffusion.tracking</code>	<code>cmp.stages.diffusion</code>
module, 137	module, 95
<code>cmp.bidsappmanager.stages.functional</code>	<code>cmp.stages.diffusion.diffusion</code>
module, 138	module, 95
<code>cmp.bidsappmanager.stages.functional.functionalMRI</code>	<code>cmp.stages.diffusion.reconstruction</code>
module, 138	module, 98
<code>cmp.bidsappmanager.stages.parcellation</code>	<code>cmp.stages.diffusion.tracking</code>
module, 139	module, 100
<code>cmp.bidsappmanager.stages.parcellation.parcellation</code>	<code>cmp.stages.functional</code>
module, 139	module, 103
<code>cmp.bidsappmanager.stages.preprocessing</code>	<code>cmp.stages.functional.functionalMRI</code>
module, 139	module, 103
<code>cmp.bidsappmanager.stages.preprocessing.fMRI_preprocessing</code>	<code>cmp.stages.parcellation</code>
module, 139	module, 104
<code>cmp.bidsappmanager.stages.preprocessing.parcellation</code>	<code>cmp.stages.parcellation.parcellation</code>
module, 140	module, 104
<code>cmp.bidsappmanager.stages.registration</code>	<code>cmp.stages.preprocessing</code>
module, 141	module, 106
<code>cmp.bidsappmanager.stages.registration.registration</code>	<code>cmp.stages.preprocessing.fmri_preprocessing</code>
module, 141	module, 106
<code>cmp.bidsappmanager.stages.segmentation</code>	<code>cmp.stages.preprocessing.preprocessing</code>
module, 142	module, 107
<code>cmp.bidsappmanager.stages.segmentation.segmentation</code>	<code>cmp.stages.registration</code>
module, 142	module, 109
<code>cmp.parser</code>	<code>cmp.stages.registration.registration</code>
module, 79	module, 109
<code>cmp.pipelines.anatomical</code>	<code>cmp.stages.segmentation</code>
module, 86	module, 113
<code>cmp.pipelines.anatomical.anatomical</code>	<code>cmp.stages.segmentation.segmentation</code>
module, 86	module, 113
<code>cmp.pipelines.common</code>	<code>CMP_BIDSAppWindow</code> (class in <i>cmp.bidsappmanager.gui</i> ), 117
module, 85	<code>CMP_BIDSAppWindowHandler</code> (class in <i>cmp.bidsappmanager.project</i> ), 125
<code>cmp.pipelines.diffusion</code>	<code>CMP_ConfigQualityWindowHandler</code> (class in <i>cmp.bidsappmanager.project</i> ), 126
module, 88	<code>CMP_ConfiguratorWindow</code> (class in <i>cmp.bidsappmanager.gui</i> ), 120
<code>cmp.pipelines.diffusion.diffusion</code>	<code>CMP_InspectorWindow</code> (class in <i>cmp.bidsappmanager.gui</i> ), 121
module, 88	<code>CMP_MainWindow</code> (class in <i>cmp.bidsappmanager.gui</i> ), 122
<code>cmp.pipelines.functional</code>	<code>CMP_MainWindowHandler</code> (class in <i>cmp.bidsappmanager.project</i> ), 127
module, 90	<code>CMP_Project_Info</code> (class in <i>cmp.project</i> ), 80
<code>cmp.pipelines.functional.eeg</code>	<code>CMP_Project_InfoUI</code> (class in <i>cmp.bidsappmanager.gui</i> ), 123
module, 90	<code>CMTK_cmat</code> , 199
<code>cmp.pipelines.functional.fMRI</code>	<code>cmtklib</code>
module, 90	module, 143
<code>cmp.project</code>	
module, 80	
<code>cmp.stages</code>	
module, 92	
<code>cmp.stages.common</code>	
module, 115	
<code>cmp.stages.connectome</code>	
module, 92	

cmtklib.bids  
     module, 143  
 cmtklib.bids.utils  
     module, 143  
 cmtklib.config  
     module, 195  
 cmtklib.connectome  
     module, 199  
 cmtklib.diffusion  
     module, 202  
 cmtklib.functionalMRI  
     module, 208  
 cmtklib.interfaces  
     module, 143  
 cmtklib.interfaces.afni  
     module, 143  
 cmtklib.interfaces.ants  
     module, 146  
 cmtklib.interfaces.camino  
     module, 147  
 cmtklib.interfaces.camino2trackvis  
     module, 150  
 cmtklib.interfaces.diffusion\_toolkit  
     module, 152  
 cmtklib.interfaces.dipy  
     module, 155  
 cmtklib.interfaces.freesurfer  
     module, 161  
 cmtklib.interfaces.fsl  
     module, 165  
 cmtklib.interfaces.misc  
     module, 174  
 cmtklib.interfaces.mrtrix3  
     module, 175  
 cmtklib.parcellation  
     module, 212  
 cmtklib.util  
     module, 220  
 CombineParcellations, 212  
 compute\_curvature  
     (*cmp.stages.connectome.connectome.ConnectomeConfig*  
     attribute), 92  
 compute\_curvature\_array() (in module *cmtk-*  
     *lib.connectome*), 200  
 compute\_length\_array() (in module *cmtk-*  
     *lib.diffusion*), 207  
 ComputeParcellationRoiVolumes, 213  
 ConcatOutputsAsTuple, 174  
 config (*cmp.stages.common.Stage* attribute), 116  
 config\_view (*cmp.bidsappmanager.stages.connectome.connectome.ConnectomeConfig*  
     attribute), 135  
 config\_view (*cmp.bidsappmanager.stages.connectome.fMRI.fMRIPipeline*  
     attribute), 135  
 config\_view (*cmp.bidsappmanager.stages.diffusion.diffusion.DiffusionS*  
     attribute), 136  
 config\_view (*cmp.bidsappmanager.stages.functional.functionalMRI.Fu*  
     attribute), 138  
 config\_view (*cmp.bidsappmanager.stages.parcellation.parcellation.Par*  
     attribute), 139  
 config\_view (*cmp.bidsappmanager.stages.preprocessing.fMRI\_preproce*  
     attribute), 140  
 config\_view (*cmp.bidsappmanager.stages.preprocessing.preprocessing*  
     attribute), 141  
 config\_view (*cmp.bidsappmanager.stages.registration.registration.Reg*  
     attribute), 141  
 config\_view (*cmp.bidsappmanager.stages.segmentation.segmentation.S*  
     attribute), 142  
 configurator (*cmp.bidsappmanager.gui.CMP\_MainWindow*  
     attribute), 122  
 connectivity\_metrics  
     (*cmp.bidsappmanager.stages.connectome.connectome.Connectome*  
     attribute), 134  
 connectivity\_metrics  
     (*cmp.stages.connectome.connectome.ConnectomeConfig*  
     attribute), 92  
 connectome (*cmp.bidsappmanager.pipelines.diffusion.diffusion.Diffusion*  
     attribute), 132  
 connectome (*cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipeli*  
     attribute), 133  
 ConnectomeConfig (class in  
     *cmp.stages.connectome.connectome*), 92  
 ConnectomeConfig (class in  
     *cmp.stages.connectome.fMRI\_connectome*),  
     93  
 ConnectomeConfigUI (class in  
     *cmp.bidsappmanager.stages.connectome.connectome*),  
     134  
 ConnectomeConfigUI (class in  
     *cmp.bidsappmanager.stages.connectome.fMRI\_connectome*),  
     135  
 ConnectomeStage (class in  
     *cmp.stages.connectome.connectome*), 92  
 ConnectomeStage (class in  
     *cmp.stages.connectome.fMRI\_connectome*),  
     94  
 ConnectomeStageUI (class in  
     *cmp.bidsappmanager.stages.connectome.connectome*),  
     134  
 ConnectomeStageUI (class in  
     *cmp.bidsappmanager.stages.connectome.fMRI\_connectome*),  
     135  
 ConstrainedSphericalDeconvolution, 177  
 copyFileToFreeSurfer (*cmp.stages.connectome.connectome.ConnectomeStageUI*  
     attribute), 111  
 copyFileToFreeSurfer, 164  
 create\_annot\_label() (in module *cmtk-*

`lib.parcellation)`, 217  
`create_dipy_recon_flow()` (in module `cmp.stages.diffusion.reconstruction`), 100  
`create_dipy_tracking_flow()` (in module `cmp.stages.diffusion.tracking`), 102  
`create_endpoints_array()` (in module `cmklib.connectome`), 200  
`create_mrtrix_recon_flow()` (in module `cmp.stages.diffusion.reconstruction`), 100  
`create_mrtrix_tracking_flow()` (in module `cmp.stages.diffusion.tracking`), 102  
`create_pipeline_flow()` (`cmp.pipelines.anatomical.anatomical.AnatomicalPipeline` attribute), 87  
`create_pipeline_flow()` (`cmp.pipelines.diffusion.diffusion.DiffusionPipeline` method), 88  
`create_pipeline_flow()` (`cmp.pipelines.functional.fMRI.fMRIPipeline` method), 91  
`create_results_plkz_local()` (in module `cmklib.util`), 220  
`create_roi()` (in module `cmklib.parcellation`), 217  
`create_roi_v2()` (in module `cmklib.parcellation`), 217  
`create_stage_flow()` (`cmp.pipelines.common.Pipeline` method), 85  
`create_T1_and_Brain()` (in module `cmklib.parcellation`), 217  
`create_view()` (`cmp.bidsappmanager.gui.CMP_Project_InfoUI` attribute), 124  
`create_wm_mask()` (in module `cmklib.parcellation`), 218  
`create_wm_mask_v2()` (in module `cmklib.parcellation`), 218  
`create_workflow()` (`cmp.stages.connectome.connectome.ConnectomeStage` method), 93  
`create_workflow()` (`cmp.stages.connectome.fmri_connectome.ConnectomeStage` method), 94  
`create_workflow()` (`cmp.stages.diffusion.diffusion.DiffusionStage` method), 96, 97  
`create_workflow()` (`cmp.stages.functional.functionalMRI.FunctionalMRIStage` method), 103, 104  
`create_workflow()` (`cmp.stages.parcellation.parcellation.ParcellationStage` method), 105  
`create_workflow()` (`cmp.stages.preprocessing.fmri_preprocessing.PreprocessingStage` method), 106, 107  
`create_workflow()` (`cmp.stages.preprocessing.preprocessing.PreprocessingStage` method), 108, 109  
`create_workflow()` (`cmp.stages.registration.registration.RegistrationStage` method), 112  
`create_workflow()` (`cmp.stages.segmentation.segmentation.SegmentationStage` method), 114, 115  
`CreateAcqpFile`, 167  
`CreateIndexFile`, 167  
`creation_mode` (`cmp.bidsappmanager.gui.CMP_Project_InfoUI` attribute), 123  
`crop_and_move_datasets()` (in module `cmklib.parcellation`), 218  
`crop_and_move_WM_and_GM()` (in module `cmklib.parcellation`), 218  
`crop_at_gmwmi` (`cmp.stages.diffusion.tracking.MRtrix_tracking_config` attribute), 102  
`CSD`, 155  
`csf` (`cmp.stages.functional.functionalMRI.FunctionalMRIConfig` attribute), 103  
`custom_track_file` (`cmp.stages.diffusion.diffusion.DiffusionConfig` attribute), 95  
`cutoff_value` (`cmp.stages.diffusion.tracking.MRtrix_tracking_config` attribute), 102  

## D

`data_provenance_tracking` (`cmp.bidsappmanager.gui.CMP_BIDSAppWindow` attribute), 118  
`datalad_dataset_path` (`cmp.bidsappmanager.gui.CMP_Project_InfoUI` attribute), 123  
`datalad_is_available` (`cmp.bidsappmanager.gui.CMP_BIDSAppWindow` attribute), 118  
`datalad_update_environment` (`cmp.bidsappmanager.gui.CMP_BIDSAppWindow` attribute), 118  
`dataset_view` (`cmp.bidsappmanager.gui.CMP_Project_InfoUI` attribute), 123  
`define_atlas_variables()` (in module `cmklib.parcellation`), 218  
`define_custom_mapping()` (`cmp.pipelines.anatomical.anatomical.AnatomicalPipeline` method), 87  
`define_custom_mapping()` (`cmp.pipelines.diffusion.diffusion.DiffusionPipeline` method), 89  
`define_custom_mapping()` (`cmp.pipelines.functional.fMRI.fMRIPipeline` method), 91

define\_inspect\_outputs() (cmp.stages.connectome.connectome.ConnectomeStage attribute), 95  
 (cmp.stages.connectome.connectome.ConnectomeStage method), 93  
 define\_inspect\_outputs() (cmp.stages.connectome.fmri\_connectome.ConnectomeStage attribute), 94  
 (cmp.stages.connectome.fmri\_connectome.ConnectomeStage method), 94  
 define\_inspect\_outputs() (cmp.stages.diffusion.diffusion.DiffusionStage attribute), 96  
 (cmp.stages.diffusion.diffusion.DiffusionStage method), 97  
 define\_inspect\_outputs() (cmp.stages.functional.functionalMRI.FunctionalMRIStage attribute), 95  
 (cmp.stages.functional.functionalMRI.FunctionalMRIStage method), 104  
 define\_inspect\_outputs() (cmp.stages.parcellation.parcellation.ParcellationStage attribute), 136  
 (cmp.stages.parcellation.parcellation.ParcellationStage method), 105  
 define\_inspect\_outputs() (cmp.stages.preprocessing.fmri\_preprocessing.PreprocessingStage attribute), 107  
 (cmp.stages.preprocessing.fmri\_preprocessing.PreprocessingStage method), 107  
 define\_inspect\_outputs() (cmp.stages.preprocessing.preprocessing.PreprocessingStage attribute), 96  
 (cmp.stages.preprocessing.preprocessing.PreprocessingStage method), 109  
 define\_inspect\_outputs() (cmp.stages.registration.registration.RegistrationStage attribute), 136  
 (cmp.stages.registration.registration.RegistrationStage method), 112  
 define\_inspect\_outputs() (cmp.stages.segmentation.segmentation.SegmentationStage attribute), 95  
 (cmp.stages.segmentation.segmentation.SegmentationStage method), 115  
 denoising (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 107  
 denoising\_algo (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 107  
 description (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 107  
 desired\_number\_of\_tracks (cmp.stages.diffusion.tracking.MRtrix\_tracking\_config attribute), 101  
 Despiking, 145  
 despiking (cmp.stages.preprocessing.fmri\_preprocessing.PreprocessingConfig attribute), 106  
 Detrending, 208  
 DiffUnpack, 153  
 diffusion (cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipeline attribute), 132  
 diffusion\_imaging\_model (cmp.project.CMP\_Project\_Info attribute), 80  
 diffusion\_imaging\_model (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 95  
 diffusion\_imaging\_model (cmp.stages.registration.registration.RegistrationConfig attribute), 109  
 diffusion\_imaging\_model\_editor (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 96  
 diffusion\_model\_editor (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 96  
 DiffusionConfig (class in cmp.stages.diffusion.diffusion), 95  
 DiffusionConfigUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 136  
 DiffusionPipeline (class in cmp.pipelines.diffusion.diffusion), 88  
 DiffusionPipelineUI (class in cmp.bidsappmanager.pipelines.diffusion.diffusion), 131  
 DiffusionStage (class in cmp.stages.diffusion.diffusion), 96  
 DiffusionStageUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 136  
 dilate\_rois (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 95  
 dilation\_kernel (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 95  
 dilation\_radius (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 95  
 dipy\_noise\_model (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 107  
 Dipy\_recon\_config (class in cmp.stages.diffusion.reconstruction), 98  
 dipy\_recon\_config (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 96  
 Dipy\_tracking\_config (class in cmp.bidsappmanager.pipelines.diffusion.tracking), 100  
 Dipy\_tracking\_config (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 96  
 Dipy\_tracking\_configUI (class in cmp.bidsappmanager.stages.diffusion.tracking), 137  
 DirectionGetterTractography, 157  
 discard\_n\_volumes (cmp.stages.preprocessing.fmri\_preprocessing.PreprocessingConfig attribute), 106  
 Discard\_tp, 208  
 dmri\_available (cmp.project.CMP\_Project\_Info attribute), 80



dmri\_available\_config (cmp.project.CMP\_Project\_Info attribute), 82  
 dmri\_bids\_acq (cmp.pipelines.diffusion.diffusion.Global\_Configuration attribute), 90  
 dmri\_bids\_acq (cmp.project.CMP\_Project\_Info attribute), 80  
 dmri\_bids\_acq\_view (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 124  
 dmri\_bids\_acqs (cmp.project.CMP\_Project\_Info attribute), 80  
 dmri\_config (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow attribute), 118  
 dmri\_config\_error\_msg (cmp.project.CMP\_Project\_Info attribute), 82  
 dmri\_config\_error\_view (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 124  
 dmri\_config\_to\_load (cmp.project.CMP\_Project\_Info attribute), 82  
 dmri\_custom\_last\_stage (cmp.project.CMP\_Project\_Info attribute), 82  
 dmri\_inputs\_checked (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow attribute), 118  
 dmri\_inputs\_checked (cmp.bidsappmanager.gui.CMP\_InspectorWindow attribute), 121  
 dmri\_inputs\_checked (cmp.bidsappmanager.project.CMP\_ConfigQualityWindow attribute), 126  
 dmri\_inputs\_checked (cmp.bidsappmanager.project.CMP\_MainWindow attribute), 128  
 dmri\_load\_config\_ini() (in module cmtklib.config), 196  
 dmri\_load\_config\_json() (in module cmtklib.config), 196  
 dmri\_pipeline (cmp.bidsappmanager.gui.CMP\_ConfiguratorWindow attribute), 120  
 dmri\_pipeline (cmp.bidsappmanager.gui.CMP\_InspectorWindow attribute), 121  
 dmri\_pipeline (cmp.bidsappmanager.gui.CMP\_MainWindow attribute), 122  
 dmri\_pipeline (cmp.bidsappmanager.project.CMP\_ConfigQualityWindow attribute), 126  
 dmri\_pipeline (cmp.bidsappmanager.project.CMP\_MainWindow attribute), 128  
 dmri\_processed (cmp.bidsappmanager.project.CMP\_MainWindow attribute), 126  
 dmri\_run (cmp.project.CMP\_Project\_Info attribute), 81  
 dmri\_runs (cmp.project.CMP\_Project\_Info attribute), 80  
 dmri\_save\_config (cmp.bidsappmanager.gui.CMP\_ConfiguratorWindow attribute), 120  
 dmri\_save\_config() (in module cmtklib.config), 196  
 dmri\_select\_config\_to\_load (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 125  
 dmri\_stage\_names (cmp.project.CMP\_Project\_Info attribute), 82  
 dmri\_warning\_view (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 124  
 docker\_process (cmp.bidsappmanager.project.CMP\_BIDSAppWindow attribute), 125  
 dof (cmp.stages.registration.registration.RegistrationConfig attribute), 111  
 DTIEstimateResponseSH, 156  
 DTIRecon, 152  
 DTLUTGen, 147  
 DVARS\_thr (cmp.stages.connectome.fnri\_connectome.ConnectomeConfig attribute), 93  
 DWI2Tensor, 178  
 DWIBiasCorrect, 179  
 DWIDenoise, 180  
**E**  
 Eddy, 167  
 eddy\_current\_and\_motion\_correction (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 108  
 eddy\_correction\_algo (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 108  
 eddy\_current\_and\_motion\_correction (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 108  
 eddy\_current\_and\_motion\_correction (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 108  
 eddy\_mask() (in module cmtklib.parcellation), 218  
 EstimateResponseForSH, 182  
 extract\_base\_directory() (in module cmp.stages.segmentation.segmentation), 115  
 extract\_freesurfer\_subject\_dir() (in module cmp.stages.segmentation.segmentation), 115  
 extract\_freesurfer\_subject\_dir() (in module cmp.stages.segmentation.segmentation), 115

ExtractFSLGrad, 183

ExtractHeaderVoxel2WorldMatrix, 174

ExtractImageVoxelSizes, 174

ExtractMRTrIXGrad, 184

ExtractPVEsFrom5TT, 202

## F

fa\_thresh(*cmp.stages.diffusion.tracking.Dipy\_tracking\_config*  
attribute), 101

FAIL(*cmklib.util.BColors* attribute), 220

fast\_use\_priors(*cmp.stages.preprocessing.preprocessing.PreprocessingConfig*  
attribute), 108

FD\_thr(*cmp.stages.connectome.fmri\_connectome.ConnectomeConfig*  
attribute), 93

fill\_stages\_outputs()  
(*cmp.pipelines.common.Pipeline* method),  
85

filter\_fibers() (in module *cmklib.diffusion*), 207

FilterTractogram, 184

flip\_table\_axis(*cmp.bidsappmanager.stages.diffusion.reconstruction.Dipy\_recon\_configUI*  
attribute), 137

flip\_table\_axis(*cmp.bidsappmanager.stages.diffusion.reconstruction.MRtrix\_recon\_configUI*  
attribute), 137

flip\_table\_axis(*cmp.stages.diffusion.reconstruction.Dipy\_recon\_config*  
attribute), 98

flip\_table\_axis(*cmp.stages.diffusion.reconstruction.MRtrix\_recon\_config*  
attribute), 99

FlipBvec, 203

FlipTable, 203

flirt\_args(*cmp.stages.registration.registration.RegistrationConfig*  
attribute), 111

fmri\_available(*cmp.project.CMP\_Project\_Info* attribute), 81

fmri\_available\_config  
(*cmp.project.CMP\_Project\_Info* attribute),  
82

fmri\_config(*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow*  
attribute), 118

fmri\_config\_error\_msg  
(*cmp.project.CMP\_Project\_Info* attribute),  
82

fmri\_config\_error\_view  
(*cmp.bidsappmanager.gui.CMP\_Project\_InfoUI*  
attribute), 124

fmri\_config\_to\_load  
(*cmp.project.CMP\_Project\_Info* attribute),  
82

fmri\_custom\_last\_stage  
(*cmp.project.CMP\_Project\_Info* attribute),  
82

fmri\_inputs\_checked  
(*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow*  
attribute), 118

fmri\_inputs\_checked

(*cmp.bidsappmanager.gui.CMP\_InspectorWindow*  
attribute), 121

fmri\_inputs\_checked

(*cmp.bidsappmanager.project.CMP\_ConfigQualityWindowHandler*  
attribute), 126

fmri\_inputs\_checked

(*cmp.bidsappmanager.project.CMP\_MainWindowHandler*  
attribute), 128

fmri\_load\_config\_ini() (in module *cmklib.config*), 196

fmri\_load\_config\_json() (in module *cmklib.config*), 196

fmri\_pipeline(*cmp.bidsappmanager.gui.CMP\_ConfiguratorWindow*  
attribute), 120

fmri\_pipeline(*cmp.bidsappmanager.gui.CMP\_InspectorWindow*  
attribute), 121

fmri\_pipeline(*cmp.bidsappmanager.gui.CMP\_MainWindow*  
attribute), 122

fmri\_pipeline(*cmp.bidsappmanager.project.CMP\_ConfigQualityWindowHandler*  
attribute), 126

fmri\_pipeline(*cmp.bidsappmanager.project.CMP\_MainWindowHandler*  
attribute), 128

fmri\_processed(*cmp.bidsappmanager.project.CMP\_ConfigQualityWindowHandler*  
attribute), 126

fmri\_processed(*cmp.bidsappmanager.project.CMP\_MainWindowHandler*  
attribute), 128

fmri\_run(*cmp.project.CMP\_Project\_Info* attribute),  
81

fmri\_runs(*cmp.project.CMP\_Project\_Info* attribute),  
81

fmri\_save\_config(*cmp.bidsappmanager.gui.CMP\_ConfiguratorWindow*  
attribute), 121

fmri\_save\_config() (in module *cmklib.config*),  
196

fmri\_select\_config\_to\_load  
(*cmp.bidsappmanager.gui.CMP\_Project\_InfoUI*  
attribute), 125

fmri\_stage\_names(*cmp.project.CMP\_Project\_Info*  
attribute), 82

fmri\_warning\_view  
(*cmp.bidsappmanager.gui.CMP\_Project\_InfoUI*  
attribute), 124

fMRIPipeline (class in  
*cmp.pipelines.functional.fMRI*), 90

fMRIPipelineUI (class in  
*cmp.bidsappmanager.pipelines.functional.fMRI*),  
133

force\_decode() (in module *cmklib.util*), 221

freesurfer\_args(*cmp.stages.segmentation.segmentation.SegmentationConfig*  
attribute), 114

freesurfer\_subject\_id  
(*cmp.project.CMP\_Project\_Info* attribute),  
81

freesurfer\_subject\_id

(cmp.stages.segmentation.segmentation.SegmentationConfig.process\_detail\_json() (in module  
 attribute), 114 cmtklib.config), 198  
 freesurfer\_subject\_id\_trait get\_freesurfer\_parcellation() (in module  
 (cmp.stages.segmentation.segmentation.SegmentationConfigcmp.stages.diffusion.tracking), 102  
 attribute), 114 get\_icon() (in module cmp.bidsappmanager.gui), 125  
 freesurfer\_subjects\_dir get\_node\_dictionary\_outputs() (in module  
 (cmp.project.CMP\_Project\_Info attribute), cmtklib.util), 221  
 81 get\_parcellation() (in module cmtk-  
 freesurfer\_subjects\_dir lib.parcellation), 219  
 (cmp.stages.segmentation.segmentation.SegmentationConfigget\_pipeline\_dictionary\_outputs() (in  
 attribute), 114 module cmtklib.util), 221  
 fs\_file(cmp.bidsappmanager.gui.CMP\_BIDSAppWindowget\_process\_detail\_ini() (in module cmtk-  
 attribute), 117 lib.config), 198  
 fs\_subject\_id(cmp.stages.registration.registration.RegistrationStageget\_process\_detail\_json() (in module cmtk-  
 attribute), 112 lib.config), 198  
 fs\_subjects\_dir(cmp.stages.registration.registration.RegistrationStage(cmp.pipelines.anatomical.anatomical.AnatomicalPipeline  
 attribute), 112 attribute), 87  
 fsl\_cost(cmp.stages.registration.registration.RegistrationConfigglobal\_conf(cmp.pipelines.diffusion.diffusion.DiffusionPipeline  
 attribute), 111 attribute), 89  
 FSLCreateHD, 169 global\_conf(cmp.pipelines.functional.fMRI.fMRIPipeline  
 functionalMRI(cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipelineUI  
 attribute), 133 Global\_Configuration (class in  
 FunctionalMRIConfig (class in cmp.pipelines.anatomical.anatomical), 87  
 cmp.stages.functional.functionalMRI), 103 Global\_Configuration (class in  
 FunctionalMRIConfigUI (class in cmp.pipelines.diffusion.diffusion), 89  
 cmp.bidsappmanager.stages.functional.functionalMRI), 138 Global\_Configuration (class in  
 138 cmp.pipelines.functional.fMRI), 90  
 FunctionalMRIStage (class in global\_nuisance(cmp.stages.functional.functionalMRI.FunctionalMRI  
 cmp.stages.functional.functionalMRI), 103 attribute), 103  
 FunctionalMRIStageUI (class in group\_analysis\_sconn() (in module cmtk-  
 cmp.bidsappmanager.stages.functional.functionalMRI), lib.connectome), 201  
 138

## G

`gen_outputfilelist()` (*cmtklib.diffusion.Make\_Seeds method*), 205

`Generate5tt`, 185

`generate_single_parcellation()` (*in module cmklib.parcellation*), 219

`generate_WM_and_GM_mask()` (*in module cmklib.parcellation*), 219

`GenerateGMWMIInterface`, 186

`get()` (*in module cmp.parser*), 79

`get_anat_process_detail_ini()` (*in module cmklib.config*), 197

`get_anat_process_detail_json()` (*in module cmklib.config*), 197

`get_dmri_process_detail_ini()` (*in module cmklib.config*), 197

`get_dmri_process_detail_json()` (*in module cmklib.config*), 197

`get_fmri_process_detail_ini()` (*in module cmklib.config*), 197

## H

HARDIMat, 154

has\_run() (cmp.stages.connectome.connectome.ConnectomeStage method), 93

has\_run() (cmp.stages.connectome.fmri\_connectome.ConnectomeStage method), 94

has\_run() (cmp.stages.diffusion.diffusion.DiffusionStage method), 97

has\_run() (cmp.stages.functional.functionalMRI.FunctionalMRIStage method), 104

has\_run() (cmp.stages.parcellation.parcellation.ParcellationStage method), 106

has\_run() (cmp.stages.preprocessing.fmri\_preprocessing.Preprocessing method), 107

has\_run() (cmp.stages.preprocessing.preprocessing.PreprocessingStage method), 109

has\_run() (cmp.stages.registration.registration.RegistrationStage method), 113

has\_run() (cmp.stages.segmentation.segmentation.SegmentationStage method), 115

HEADER (cmtklib.util.BColors attribute), 220



[inspect\\_outputs](#) ([cmp.stages.common.Stage](#) attribute), 116  
[imaging\\_model](#) ([cmp.pipelines.functional.fMRI.Global\\_Configuration](#) attribute), 90  
[imaging\\_model](#) ([cmp.stages.diffusion.reconstruction.Dipy\\_recon\\_config](#) attribute), 98  
[imaging\\_model](#) ([cmp.stages.diffusion.tracking.Dipy\\_tracking\\_config](#) attribute), 100  
[include\\_thalamic\\_nuclei\\_parcellation](#) ([cmp.stages.parcellation.parcellation.ParcellationConfig](#) attribute), 104  
[init](#) ([cmp.stages.registration.registration.RegistrationConfig](#) attribute), 111  
[init\\_anat\\_project\(\)](#) (in module [cmp.bidsappmanager.project](#)), 129  
[init\\_anat\\_project\(\)](#) (in module [cmp.project](#)), 82  
[init\\_dmri\\_project\(\)](#) (in module [cmp.project](#)), 83  
[init\\_fmri\\_project\(\)](#) (in module [cmp.project](#)), 83  
[input\\_folders](#) ([cmp.pipelines.anatomical.anatomical.AnatomicalPipeline](#) attribute), 87  
[input\\_folders](#) ([cmp.pipelines.diffusion.diffusion.DiffusionPipeline](#) attribute), 89  
[input\\_folders](#) ([cmp.pipelines.functional.fMRI.fMRIPipeline](#) attribute), 91  
[inspect\\_output\\_button](#) ([cmp.bidsappmanager.stages.connectome.connectome.ConnectomeStageUI](#) attribute), 134  
[inspect\\_output\\_button](#) ([cmp.bidsappmanager.stages.connectome.fmri\\_connectome.ConnectomeStageUI](#) attribute), 135  
[inspect\\_output\\_button](#) ([cmp.bidsappmanager.stages.diffusion.diffusion.DiffusionStageUI](#) attribute), 136  
[inspect\\_output\\_button](#) ([cmp.bidsappmanager.stages.functional.functionalMRI.FunctionalMRIStageUI](#) attribute), 138  
[inspect\\_output\\_button](#) ([cmp.bidsappmanager.stages.parcellation.parcellation.ParcellationStageUI](#) attribute), 139  
[inspect\\_output\\_button](#) ([cmp.bidsappmanager.stages.preprocessing.fmri\\_preprocessing.PreprocessingStageUI](#) attribute), 140  
[inspect\\_output\\_button](#) ([cmp.bidsappmanager.stages.preprocessing.preprocessing.PreprocessingStageUI](#) attribute), 141  
[inspect\\_output\\_button](#) ([cmp.bidsappmanager.stages.registration.registration.RegistrationStageUI](#) attribute), 141  
[inspect\\_output\\_button](#) ([cmp.bidsappmanager.stages.segmentation.segmentation.SegmentationStageUI](#) attribute), 142  
[install\\_data\\_lad\\_data\\_set\\_via\\_ssh](#) ([cmp.bidsappmanager.gui.CMP\\_Project\\_InfoUI](#) attribute), 123  
[interpolation](#) ([cmp.stages.preprocessing.preprocessing.PreprocessingStageUI](#) attribute), 108  
[is\\_running\(\)](#) ([cmp.stages.common.Stage](#) method), 116  
[is\\_tool\(\)](#) (in module [cmp.bidsappmanager.project](#)), 129  
[ismember\(\)](#) ([cmp.lib.parcellation.CombineParcellations](#) method), 213  
[isotropic\\_interpolation](#) ([cmp.stages.segmentation.segmentation.SegmentationConfig](#) attribute), 113  
[isotropic\\_vox\\_size](#) ([cmp.stages.segmentation.segmentation.SegmentationConfig](#) attribute), 113  
[laplacian\\_regularization](#) ([cmp.stages.diffusion.reconstruction.Dipy\\_recon\\_config](#) attribute), 98  
[laplacian\\_weighting](#) ([cmp.stages.diffusion.reconstruction.Dipy\\_recon\\_config](#) attribute), 98

(cmp.stages.diffusion.reconstruction.Dipy\_recon\_config\_isotropic (cmp.stages.segmentation.segmentation.Segmentation attribute), 98  
 attribute), 113  
 launch\_process() (cmp.pipelines.common.Pipeline Make\_Mrtrix\_Seeds, 204  
 method), 85 Make\_Seeds, 205  
 length() (in module cmtklib.util), 221 manage\_bidsapp\_procs()  
 list\_of\_subjects\_to\_be\_processed (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow class method), 119  
 (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow attribute), 117 manage\_bidsapp\_procs()  
 lmax\_order (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config (cmp.bidsappmanager.project.CMP\_BIDSAppWindowHandler class method), 125  
 attribute), 98  
 lmax\_order (cmp.stages.diffusion.reconstruction.MRtrix\_recon\_config (cmp.bidsappmanager.gui.CMP\_MainWindow attribute), 100  
 attribute), 100  
 load\_anat\_config\_file() MAPMRI, 158  
 (cmp.bidsappmanager.project.CMP\_ConfigQualityWindowHandler (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config attribute), 98  
 method), 127  
 load\_dataset (cmp.bidsappmanager.gui.CMP\_MainWindow MainWindowCommand, 170  
 attribute), 122 max\_angle (cmp.stages.diffusion.tracking.Dipy\_tracking\_config attribute), 101  
 load\_dataset() (cmp.bidsappmanager.project.CMP\_MainWindowHandler method), 128  
 load\_dmri\_config\_file() mean\_curvature() (in module cmtklib.util), 222  
 (cmp.bidsappmanager.project.CMP\_ConfigQualityWindowHandler attribute), 90  
 method), 127 module  
 load\_fmri\_config\_file() cmp, 79  
 (cmp.bidsappmanager.project.CMP\_ConfigQualityWindowHandler cmp.bidsappmanager.core, 116  
 method), 127 cmp.bidsappmanager.gui, 117  
 load\_graphs() (in module cmtklib.util), 222 cmp.bidsappmanager.pipelines.anatomical,  
 load\_project() (cmp.bidsappmanager.project.CMP\_ConfigQualityWindowHandler cmp.bidsappmanager.pipelines.anatomical.anatomical,  
 method), 127 cmp.bidsappmanager.pipelines.diffusion,  
 local\_model (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config cmp.bidsappmanager.pipelines.diffusion.diffusion,  
 attribute), 98 cmp.bidsappmanager.pipelines.functional,  
 local\_model (cmp.stages.diffusion.reconstruction.MRtrix\_recon\_config cmp.bidsappmanager.pipelines.functional.eeg,  
 attribute), 100 cmp.bidsappmanager.pipelines.functional.fMRI,  
 local\_model\_editor 133  
 (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config cmp.bidsappmanager.project, 125  
 attribute), 98 cmp.bidsappmanager.stages, 134  
 local\_model\_editor cmp.bidsappmanager.pipelines.functional.eeg, 132  
 (cmp.stages.diffusion.reconstruction.MRtrix\_recon\_config cmp.bidsappmanager.pipelines.functional.fMRI, 133  
 attribute), 99  
 log\_visualization 134  
 (cmp.bidsappmanager.stages.connectome.connectome.ConnectomeStageUI  
 attribute), 134  
 log\_visualization (cmp.bidsappmanager.stages.connectome.fmri\_connectome.ConnectomeStageUI  
 attribute), 135  
 log\_visualization cmp.bidsappmanager.stages.connectome.connectome, 134  
 (cmp.stages.connectome.connectome.ConnectomeConfig cmp.bidsappmanager.stages.connectome.fmri\_connectome, 135  
 attribute), 92  
 log\_visualization cmp.bidsappmanager.stages.diffusion, 136  
 (cmp.stages.connectome.fmri\_connectome.ConnectomeConfig cmp.bidsappmanager.stages.diffusion.diffusion, 137  
 attribute), 94 cmp.bidsappmanager.stages.diffusion.reconstruct, 137  
 cmp.bidsappmanager.stages.diffusion.tracking,

## M

magn() (in module cmtklib.util), 222

137  
 cmp.bidsappmanager.stages.functional, 138  
 cmp.bidsappmanager.stages.functional.functionalMRI, 138  
 cmp.bidsappmanager.stages.parcellation, 139  
 cmp.bidsappmanager.stages.parcellation.parcellation, 139  
 cmp.bidsappmanager.stages.preprocessing, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.bids, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.config, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.diffusion, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.functionalMRI, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.interfaces, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.interfaces.afni, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.interfaces.ants, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.interfaces.camino, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.interfaces.camino2trackvis, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.interfaces.diffusion\_toolkit, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.interfaces.dipy, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.interfaces.freesurfer, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.interfaces.fsl, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.interfaces.misc, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.interfaces.mrtrix3, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.parcellation, 139  
 cmp.bidsappmanager.stages.preprocessing.cmtklib.util, 139  
 cmp.bidsappmanager.stages.registration, 141  
 cmp.bidsappmanager.stages.registration.registration, 141  
 cmp.bidsappmanager.stages.segmentation, 142  
 cmp.bidsappmanager.stages.segmentation.segmentation, 142  
 cmp.parser, 79  
 cmp.pipelines.anatomical, 86  
 cmp.pipelines.anatomical.anatomical, 86  
 cmp.pipelines.common, 85  
 cmp.pipelines.diffusion, 88  
 cmp.pipelines.diffusion.diffusion, 88  
 cmp.pipelines.functional, 90  
 cmp.pipelines.functional.eeg, 90  
 cmp.pipelines.functional.fMRI, 90  
 cmp.project, 80  
 cmp.stages, 92  
 cmp.stages.common, 115  
 cmp.stages.connectome, 92  
 cmp.stages.connectome.connectome, 92  
 cmp.stages.connectome.fmri\_connectome, 93  
 cmp.stages.diffusion, 95  
 cmp.stages.diffusion.diffusion, 95  
 cmp.stages.diffusion.reconstruction, 98  
 cmp.stages.diffusion.tracking, 100  
 cmp.stages.functional, 103  
 cmp.stages.functional.functionalMRI, 103  
 cmp.stages.parcellation, 104  
 cmp.stages.parcellation.parcellation, 104  
 cmp.stages.preprocessing, 106  
 cmp.stages.preprocessing.fmri\_preprocessing, 106  
 cmp.stages.preprocessing.preprocessing, 107  
 cmp.stages.registration, 109  
 cmp.stages.registration.registration, 109  
 cmp.stages.segmentation, 113  
 cmp.stages.segmentation.segmentation, 113  
 cmtklib, 143  
 cmtklib.bids, 143  
 cmtklib.config, 195  
 cmtklib.diffusion, 202  
 cmtklib.functionalMRI, 208  
 cmtklib.interfaces, 143  
 cmtklib.interfaces.afni, 143  
 cmtklib.interfaces.ants, 146  
 cmtklib.interfaces.camino, 147  
 cmtklib.interfaces.camino2trackvis, 150  
 cmtklib.interfaces.diffusion\_toolkit, 152  
 cmtklib.interfaces.dipy, 155  
 cmtklib.interfaces.freesurfer, 161  
 cmtklib.interfaces.fsl, 165  
 cmtklib.interfaces.misc, 174  
 cmtklib.interfaces.mrtrix3, 175  
 cmtklib.parcellation, 212  
 cmtklib.util, 220  
 motion (*cmp.stages.functional.functionalMRI.FunctionalMRIConfig* attribute), 103  
 motion\_correction (*cmp.stages.preprocessing.fmri\_preprocessing.PreprocessingConfig* attribute), 106  
 MRConvert, 186  
 MRCrop, 188  
 MRThreshold, 189  
 MRTransform, 190  
 MRTrix3Base, 191  
 MRtrix\_mul, 191  
 MRtrix\_recon\_config (class in *cmp.stages.diffusion.reconstruction*), 99  
 mrtrix\_recon\_config (*cmp.stages.diffusion.diffusion.DiffusionConfig* attribute), 96  
 MRtrix\_recon\_configUI (class in *cmp.bidsappmanager.stages.diffusion.reconstruction*), 137  
 MRtrix\_tracking\_config (class in *cmp.stages.diffusion.tracking*), 101  
 mrtrix\_tracking\_config (*cmp.stages.diffusion.diffusion.DiffusionConfig* attribute), 101

- attribute), 96
- MRtrix\_tracking\_configUI (class in *cmp.bidsappmanager.stages.diffusion.tracking*), 137
- MultipleANTsApplyTransforms, 146
- MultiSelectAdapter (class in *cmp.bidsappmanager.gui*), 125
- ## N
- new\_project() (*cmp.bidsappmanager.project.CMP\_ConfigQualityWindow* attribute), 127
- no\_search(*cmp.stages.registration.registration.RegistrationConfig* attribute), 111
- now(*cmp.pipelines.anatomical.anatomical.AnatomicalPipeline* attribute), 87
- now(*cmp.pipelines.diffusion.diffusion.DiffusionPipeline* attribute), 89
- now(*cmp.pipelines.functional.fMRI.fMRIPipeline* attribute), 91
- Nuisance\_regression, 209
- number\_of\_cores(*cmp.pipelines.common.Pipeline* attribute), 85
- number\_of\_cores(*cmp.project.CMP\_Project\_Info* attribute), 82
- number\_of\_parallel\_procs\_updated() (*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow* method), 119
- number\_of\_participants\_processed\_in\_parallel(*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow* attribute), 117
- number\_of\_seeds(*cmp.stages.diffusion.tracking.Dipy\_tracking\_config* attribute), 101
- number\_of\_threads(*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow* attribute), 117
- number\_of\_threads(*cmp.stages.segmentation.segmentation.SegmentationConfig* attribute), 114
- number\_threads\_max(*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow* attribute), 117
- ## O
- OKBLUE (*cmklb.util.BColors* attribute), 220
- OKGREEN (*cmklb.util.BColors* attribute), 220
- open\_view(*cmp.bidsappmanager.gui.CMP\_Project\_InfoUI* attribute), 124
- ordered\_stage\_list(*cmp.pipelines.anatomical.anatomical.AnatomicalPipeline* attribute), 87
- ordered\_stage\_list(*cmp.pipelines.diffusion.diffusion.DiffusionPipeline* attribute), 89
- ordered\_stage\_list(*cmp.pipelines.functional.fMRI.fMRIPipeline* attribute), 91
- Orient, 171
- output\_anat\_available(*cmp.bidsappmanager.gui.CMP\_InspectorWindow* attribute), 121
- output\_dir(*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow* attribute), 117
- output\_dir(*cmp.pipelines.common.Stage* attribute), 115
- output\_directory(*cmp.project.CMP\_Project\_Info* attribute), 80
- output\_dmri\_available(*cmp.bidsappmanager.gui.CMP\_InspectorWindow* attribute), 122
- output\_fmri\_available(*cmp.bidsappmanager.gui.CMP\_InspectorWindow* attribute), 122
- output\_types(*cmp.bidsappmanager.stages.connectome.connectome.ConnectomeConfig* attribute), 134
- output\_types(*cmp.bidsappmanager.stages.connectome.fmri\_connectome.ConnectomeConfig* attribute), 135
- output\_types(*cmp.stages.connectome.connectome.ConnectomeConfig* attribute), 92
- output\_types(*cmp.stages.connectome.fmri\_connectome.ConnectomeConfig* attribute), 93
- ## P
- Parcellate, 214
- ParcellateBrainstemStructures, 215
- ParcellateHippocampalSubfields, 215
- ParcellateThalamus, 216
- parcellation(*cmp.bidsappmanager.pipelines.anatomical.anatomical.AnatomicalPipeline* attribute), 130
- parcellation\_scheme(*cmp.project.CMP\_Project\_Info* attribute), 81
- parcellation\_scheme(*cmp.stages.parcellation.parcellation.ParcellationConfig* attribute), 104
- parcellation\_scheme\_editor(*cmp.stages.parcellation.parcellation.ParcellationConfig* attribute), 104
- ParcellationConfig (class in *cmp.stages.parcellation.parcellation*), 104
- ParcellationConfigUI (class in *cmp.bidsappmanager.stages.parcellation.parcellation*), 139
- ParcellationStage (class in *cmp.stages.parcellation.parcellation*), 105
- ParcellationStageUI (class in *cmp.bidsappmanager.stages.parcellation.parcellation*), 139
- partial\_volume\_estimation

(cmp.stages.preprocessing.preprocessing.PreprocessingConfig (cmp.pipelines.functional.fMRI.fMRIPipeline  
 attribute), 108 method), 91  
 PicoPDFs, 148 process\_type (cmp.pipelines.anatomical.anatomical.Global\_Configuration  
 Pipeline (class in cmp.pipelines.common), 85 attribute), 87  
 pipeline (cmp.pipelines.common.ProcessThread at- process\_type (cmp.pipelines.diffusion.diffusion.Global\_Configuration  
 tribute), 85 attribute), 89  
 pipeline (cmp.stages.registration.registration.RegistrationConfig process\_type (cmp.pipelines.functional.fMRI.Global\_Configuration  
 attribute), 109 attribute), 90  
 pipeline\_group (cmp.bidsappmanager.pipelines.anatomical.anatomical.AnatomicalPipelineUI  
 attribute), 130 (cmp.stages.diffusion.diffusion.DiffusionConfig  
 pipeline\_group (cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipelineUI  
 attribute), 132 ProcessThread (class in cmp.pipelines.common), 85  
 pipeline\_group (cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipelineUI (class in cmp.pipelines.common),  
 attribute), 133 85  
 pipeline\_mode (cmp.stages.parcellation.parcellation.ParcellationConfig window (class in cmp.pipelines.common),  
 attribute), 104 86  
 pipeline\_processing\_summary\_view project\_info (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow  
 (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 117  
 attribute), 123 project\_info (cmp.bidsappmanager.gui.CMP\_ConfiguratorWindow  
 positivity\_constraint attribute), 120  
 (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config project\_info (cmp.bidsappmanager.gui.CMP\_InspectorWindow  
 attribute), 98 attribute), 121  
 preprocessing (cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipelineUI (cmp.bidsappmanager.gui.CMP\_MainWindow  
 attribute), 131 attribute), 122  
 preprocessing (cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipelineUI (cmp.bidsappmanager.project.CMP\_ConfigQualityWin  
 attribute), 133 attribute), 126  
 PreprocessingConfig (class in project\_loaded (cmp.bidsappmanager.project.CMP\_MainWindowHan  
 cmp.stages.preprocessing.fmri\_preprocessing), attribute), 127  
 106 pw (cmp.pipelines.common.ProgressThread attribute), 85  
 PreprocessingConfig (class in Q  
 cmp.stages.preprocessing.preprocessing), 107  
 PreprocessingConfigUI (class in quality\_control (cmp.bidsappmanager.gui.CMP\_MainWindow  
 cmp.bidsappmanager.stages.preprocessing.fmri\_preprocessing), attribute), 122  
 139 R  
 PreprocessingConfigUI (class in radial\_order (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config  
 cmp.bidsappmanager.stages.preprocessing.preprocessing), attribute), 99  
 140 radial\_order\_values  
 PreprocessingStage (class in (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config  
 cmp.stages.preprocessing.fmri\_preprocessing), attribute), 99  
 106 recon\_mode (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config  
 PreprocessingStage (class in attribute), 98  
 cmp.stages.preprocessing.preprocessing), recon\_mode (cmp.stages.diffusion.reconstruction.MRtrix\_recon\_config  
 108 attribute), 100  
 PreprocessingStageUI (class in recon\_processing\_tool  
 cmp.bidsappmanager.stages.preprocessing.fmri\_preprocessing), (cmp.stages.diffusion.diffusion.DiffusionConfig  
 140 attribute), 95  
 PreprocessingStageUI (class in recon\_processing\_tool\_editor  
 cmp.bidsappmanager.stages.preprocessing.preprocessing), (cmp.stages.diffusion.diffusion.DiffusionConfig  
 140 attribute), 95  
 process () (cmp.pipelines.anatomical.anatomical.AnatomicalPipeline refresh\_folder () (in module  
 method), 87 cmp.bidsappmanager.project), 129  
 process () (cmp.pipelines.diffusion.diffusion.DiffusionPipeline refresh\_folder () (in module cmp.project), 83  
 method), 89



registration(*cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipelineUI* attribute), 131  
 registration(*cmp.bidsappmanager.pipelines.functional.fMRI.fMRIBispectralUI* attribute), 133  
 registration\_mode(*cmp.stages.registration.registration.RegistrationConfig* attribute), 109  
 registration\_mode\_trait(*cmp.stages.registration.registration.RegistrationConfig* attribute), 109  
 RegistrationConfig (class in *SD* (*cmp.stages.diffusion.tracking.MRtrix\_tracking\_config* attribute), 101  
 RegistrationConfigUI (class in *seed\_density* (*cmp.stages.diffusion.tracking.Dipy\_tracking\_config* attribute), 101  
 RegistrationStage (class in *seed\_from\_gmwmi* (*cmp.stages.diffusion.tracking.Dipy\_tracking\_config* attribute), 101  
 RegistrationStageUI (class in *seed\_from\_gmwmi* (*cmp.stages.diffusion.tracking.MRtrix\_tracking\_config* attribute), 102  
 repetition\_time(*cmp.stages.preprocessing.fMRI\_preprocessing.PreprocessingConfig* attribute), 106  
 resampling(*cmp.stages.preprocessing.preprocessing.PreprocessingConfig* attribute), 108  
 return\_button\_style\_sheet() (in module *cmklib.util*), 223  
 ROI\_idx (*cmklib.diffusion.Make\_Mrtrix\_Seeds* attribute), 205  
 ROI\_idx (*cmklib.diffusion.Make\_Seeds* attribute), 205  
 rsfmri\_conmat, 201  
 run() (*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow* class method), 119  
 run() (*cmp.pipelines.common.ProcessThread* method), 85  
 run() (*cmp.pipelines.common.ProgressThread* method), 85  
 run() (in module *cmp.bidsappmanager.core*), 116  
 run\_anat\_pipeline(*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow* attribute), 118  
 run\_dmri\_pipeline(*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow* attribute), 118  
 run\_fmri\_pipeline(*cmp.bidsappmanager.gui.CMP\_BIDSAppWindow* attribute), 118  
 run\_individual() (in module *cmp.project*), 84  
 S  
 save\_all\_config(*cmp.bidsappmanager.gui.CMP\_ConfiguratorWindow* attribute), 121  
 save\_anat\_config\_file() (*cmp.bidsappmanager.project.CMP\_ConfigQualityWindowHandler* class method), 127  
 save\_fibers() (in module *cmklib.connectome*), 202  
 save\_fmri\_config\_file() (*cmp.bidsappmanager.project.CMP\_ConfigQualityWindowHandler* class method), 127  
 Scrubbing, 210  
*SD* (*cmp.stages.diffusion.tracking.MRtrix\_tracking\_config* attribute), 101  
 seed\_density(*cmp.stages.diffusion.tracking.Dipy\_tracking\_config* attribute), 101  
 seed\_from\_gmwmi (*cmp.stages.diffusion.tracking.Dipy\_tracking\_config* attribute), 101  
 seed\_from\_gmwmi (*cmp.stages.diffusion.tracking.MRtrix\_tracking\_config* attribute), 102  
 seg\_tool (*cmp.stages.segmentation.segmentation.SegmentationConfig* attribute), 113  
 segmentation (*cmp.bidsappmanager.pipelines.anatomical.anatomical.A* attribute), 130  
 SegmentationConfig (class in *cmp.stages.segmentation.segmentation*), 113  
 SegmentationConfigUI (class in *cmp.bidsappmanager.stages.segmentation.segmentation*), 142  
 SegmentationStage (class in *cmp.stages.segmentation.segmentation*), 114  
 SegmentationStageUI (class in *cmp.bidsappmanager.stages.segmentation.segmentation*), 142  
 select\_subject() (*cmp.bidsappmanager.gui.CMP\_InspectorWindow* method), 122  
 sh\_order (*cmp.stages.diffusion.tracking.Dipy\_tracking\_config* attribute), 101  
 SHORE, 159  
 shore\_constrain\_e0 (*cmp.stages.diffusion.reconstruction.Dipy\_recon\_config* attribute), 99  
 shore\_lambda\_l (*cmp.stages.diffusion.reconstruction.Dipy\_recon\_config* attribute), 99  
 shore\_lambda\_n (*cmp.stages.diffusion.reconstruction.Dipy\_recon\_config* attribute), 99  
 shore\_positive\_constraint (*cmp.stages.diffusion.reconstruction.Dipy\_recon\_config* attribute), 99  
 shore\_radial\_order

(cmp.stages.diffusion.reconstruction.Dipy\_recon\_config attribute), 97  
 shore\_tau (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config attribute), 87  
 shore\_zeta (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config attribute), 85  
 show\_bidsapp\_interface() subject (cmp.project.CMP\_Project\_Info attribute), 80  
 (cmp.bidsappmanager.gui.CMP\_MainWindow method), 123 subject (cmp.stages.connectome.connectome.ConnectomeConfig attribute), 92  
 show\_bidsapp\_window() subject (cmp.stages.connectome.fmri\_connectome.ConnectomeConfig attribute), 94  
 (cmp.bidsappmanager.project.CMP\_ConfigQualityWindow class method), 127 subject\_session (cmp.pipelines.anatomical.anatomical.Global\_Configuration attribute), 87  
 sift (cmp.stages.diffusion.tracking.MRtrix\_tracking\_config attribute), 102 subject\_session (cmp.pipelines.diffusion.diffusion.Global\_Configuration attribute), 89  
 SIFT2, 192  
 single\_fib\_thr (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config attribute), 98 subject\_session (cmp.project.CMP\_Project\_Info attribute), 80  
 single\_fib\_thr (cmp.stages.diffusion.reconstruction.MRtrix\_tracking\_config attribute), 100 (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 124  
 slice\_timing (cmp.stages.preprocessing.fmri\_preprocessing.PreprocessingConfig attribute), 106 subject\_sessions (cmp.project.CMP\_Project\_Info attribute), 80  
 small\_delta (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config attribute), 99 subject\_view (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 117  
 SplitDiffusion, 205 subjects (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow attribute), 117  
 ssh\_remote (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 123 subjects (cmp.pipelines.anatomical.anatomical.Global\_Configuration attribute), 87  
 ssh\_user (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 123 subjects (cmp.pipelines.diffusion.diffusion.Global\_Configuration attribute), 89  
 Stage (class in cmp.stages.common), 115 subjects (cmp.project.CMP\_Project\_Info attribute), 80  
 stage\_names (cmp.pipelines.common.ProgressThread attribute), 86 summary\_view\_button  
 stages (cmp.pipelines.common.ProgressThread attribute), 86 (cmp.bidsappmanager.gui.CMP\_Project\_InfoUI attribute), 123  
 start\_bids\_app() (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow method), 119  
 start\_bids\_app() (cmp.bidsappmanager.project.CMP\_BIDSAppWindowHandler method), 125  
 start\_bidsapp (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow attribute), 118  
 start\_bidsapp\_participant\_level\_process (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow method), 119  
 start\_bidsapp\_participant\_level\_process\_with\_data (cmp.bidsappmanager.gui.CMP\_BIDSAppWindow method), 119  
 start\_bidsapp\_process() tracking\_mode (cmp.stages.diffusion.tracking.Dipy\_tracking\_config attribute), 100  
 (cmp.bidsappmanager.project.CMP\_BIDSAppWindowHandler class method), 125 tracking\_mode (cmp.stages.diffusion.tracking.MRtrix\_tracking\_config attribute), 101  
 step\_size (cmp.stages.diffusion.tracking.Dipy\_tracking\_config attribute), 101 tracking\_processing\_tool  
 stop\_bids\_app() (cmp.bidsappmanager.project.CMP\_BIDSAppWindowHandler class method), 126 (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 95  
 StreamlineTrack, 192 tracking\_processing\_tool  
 strip\_suffix() (in module (cmp.stages.diffusion.reconstruction.Dipy\_recon\_config

`attribute`), 98  
`tracking_processing_tool_editor`  
     (`cmp.stages.diffusion.diffusion.DiffusionConfig`  
     `attribute`), 95  
`traits_view(cmp.bidsappmanager.gui.CMP_BIDSAppWindow`  
     `attribute`), 119  
`traits_view(cmp.bidsappmanager.gui.CMP_ConfiguratorWindow`  
     `attribute`), 121  
`traits_view(cmp.bidsappmanager.gui.CMP_InspectorWindow`  
     `attribute`), 122  
`traits_view(cmp.bidsappmanager.gui.CMP_MainWindow`  
     `attribute`), 123  
`traits_view(cmp.bidsappmanager.gui.CMP_Project_InfoUI`  
     `attribute`), 124  
`traits_view(cmp.bidsappmanager.pipelines.anatomical.anatomical.AnatomicalPipelineUI)`  
     `attribute`), 131  
`traits_view(cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipelineUI)`  
     `attribute`), 132  
`traits_view(cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipelineUI)`  
     `attribute`), 133  
`traits_view(cmp.bidsappmanager.stages.connectome.connectome.ConnectomeConfigUI)`  
     `attribute`), 134  
`traits_view(cmp.bidsappmanager.stages.connectome.fmri_connectome.ConnectomeConfigUI)`  
     `attribute`), 135  
`traits_view(cmp.bidsappmanager.stages.diffusion.diffusion.DiffusionConfigUI)`  
     `attribute`), 136  
`traits_view(cmp.bidsappmanager.stages.diffusion.reconstruction.dipy_recon.configUI)`  
     `attribute`), 137  
`traits_view(cmp.bidsappmanager.stages.diffusion.reconstruction.MRtrixrecon.configUI)`  
     `attribute`), 137  
`traits_view(cmp.bidsappmanager.stages.diffusion.tracking.Dipy_tracking_configUI)`  
     `attribute`), 137  
`traits_view(cmp.bidsappmanager.stages.diffusion.tracking.MRtrix_tracking_configUI)`  
     `attribute`), 138  
`traits_view(cmp.bidsappmanager.stages.functional.functional.fMRI.FunctionalfMRIConfigUI)`  
     `attribute`), 138  
`traits_view(cmp.bidsappmanager.stages.parcellation.parcellation.ParcellationConfigUI)`  
     `attribute`), 139  
`traits_view(cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing.preprocessingDiffusionPipeline)`  
     `attribute`), 140  
`traits_view(cmp.bidsappmanager.stages.preprocessing_fmri.PreprocessingPreprocessingConfigUI)`  
     `attribute`), 140  
`traits_view(cmp.bidsappmanager.stages.registration.registration.RegistrationConfigUI)`  
     `attribute`), 141  
`traits_view(cmp.bidsappmanager.stages.segmentation.segmentation.SegmentationConfigUI)`  
     `attribute`), 142

## U

`UNDERLINE(cmtklib.util.BColors attribute)`, 220  
`unicode2str()` (in module `cmp.stages.registration.registration`), 113  
`update_anat_last_processed()` (in module `cmp.bidsappmanager.project`), 129  
`update_anat_last_processed()` (in module `cmp.project`), 84  
`update_atlas_info()`  
     (`cmp.stages.parcellation.parcellation.ParcellationConfig`  
     method), 105  
`update_checksettings()`  
     (`cmp.bidsappmanager.gui.CMP_BIDSAppWindow`  
     method), 120  
`update_diffusion_imaging_model()`  
     (`cmp.bidsappmanager.gui.CMP_ConfiguratorWindow`  
     method), 121  
`update_diffusion_imaging_model()`  
     (`cmp.bidsappmanager.gui.CMP_InspectorWindow`  
     method), 122  
`update_dipy_tracking_sh_order()`  
     (`cmp.stages.diffusion.diffusion.DiffusionConfig`  
     method), 96  
`update_run_anat_pipeline()` (in module `cmp.bidsappmanager.project`), 130  
`update_run_dmri_pipeline()` (in module `cmp.project`), 84  
`update_run_anat_pipeline()` (in module `cmp.bidsappmanager.project`), 130  
`update_run_dmri_pipeline()` (in module `cmp.project`), 84  
`update_registration()`  
     (`cmp.pipelines.functional.fMRI.fMRIPipeline`  
     method), 91  
`update_outputs_tracking()`  
     (`cmp.pipelines.diffusion.diffusion.DiffusionPipeline`  
     method), 89  
`update_registration()`  
     (`cmp.pipelines.functional.fMRI.fMRIPipeline`  
     method), 91  
`update_run_anat_pipeline()`  
     (`cmp.bidsappmanager.gui.CMP_BIDSAppWindow`  
     method), 120  
`update_run_dmri_pipeline()`  
     (`cmp.bidsappmanager.gui.CMP_BIDSAppWindow`



`method`), 120  
`update_run_fmri_pipeline()`  
     (`cmp.bidsappmanager.gui.CMP_BIDSAppWindow`  
     `method`), 120  
`update_scrubbing()`  
     (`cmp.pipelines.functional.fMRI.fMRIPipeline`  
     `method`), 92  
`update_segmentation()`  
     (`cmp.pipelines.anatomical.anatomical.AnatomicalPipeline`  
     `method`), 87  
`update_subject_anat_pipeline()`  
     (`cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler`  
     `method`), 127  
`update_subject_dmri_pipeline()`  
     (`cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler`  
     `method`), 127  
`update_subject_fmri_pipeline()`  
     (`cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler`  
     `method`), 127  
`update_vizualization_layout()`  
     (`cmp.pipelines.diffusion.diffusion.DiffusionPipeline`  
     `method`), 89  
`update_vizualization_logscale()`  
     (`cmp.pipelines.diffusion.diffusion.DiffusionPipeline`  
     `method`), 89  
`UpdateGMWMInterfaceSeeding`, 206  
`use_act` (`cmp.stages.diffusion.tracking.Dipy_tracking_config`  
     `attribute`), 101  
`use_act` (`cmp.stages.diffusion.tracking.MRtrix_tracking_config`  
     `attribute`), 102  
`use_existing_freesurfer_data`  
     (`cmp.stages.segmentation.segmentation.SegmentationConfig`  
     `attribute`), 114  
`use_float_precision`  
     (`cmp.stages.registration.registration.RegistrationConfig`  
     `attribute`), 110  
`use_fsl_brain_mask`  
     (`cmp.stages.segmentation.segmentation.SegmentationConfig`  
     `attribute`), 114  
`uses_qform` (`cmp.stages.registration.registration.RegistrationConfig`  
     `attribute`), 111

## V

`view_mode` (`cmp.bidsappmanager.gui.CMP_MainWindow`  
     `attribute`), 123  
`view_mode` (`cmp.bidsappmanager.pipelines.anatomical.anatomical.AnatomicalPipelineUI`  
     `attribute`), 130  
`view_mode` (`cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipelineUI`  
     `attribute`), 132  
`view_mode` (`cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipelineUI`  
     `attribute`), 133  
`Voxel2Image`, 149

## W

`WARNING` (`cmtklib.util.BColors` `attribute`), 220  
`white_matter_mask`  
     (`cmp.stages.segmentation.segmentation.SegmentationConfig`  
     `attribute`), 114  
`wm` (`cmp.stages.functional.fMRI.FunctionalMRICongig`  
     `attribute`), 103  
`write_derivative_description()` (in module  
     `cmtklib.bids.utils`), 143