
CMP3 Documentation

Release v3.0.0

Brain Communication Pathways Sinergia Consortium

Dec 23, 2021

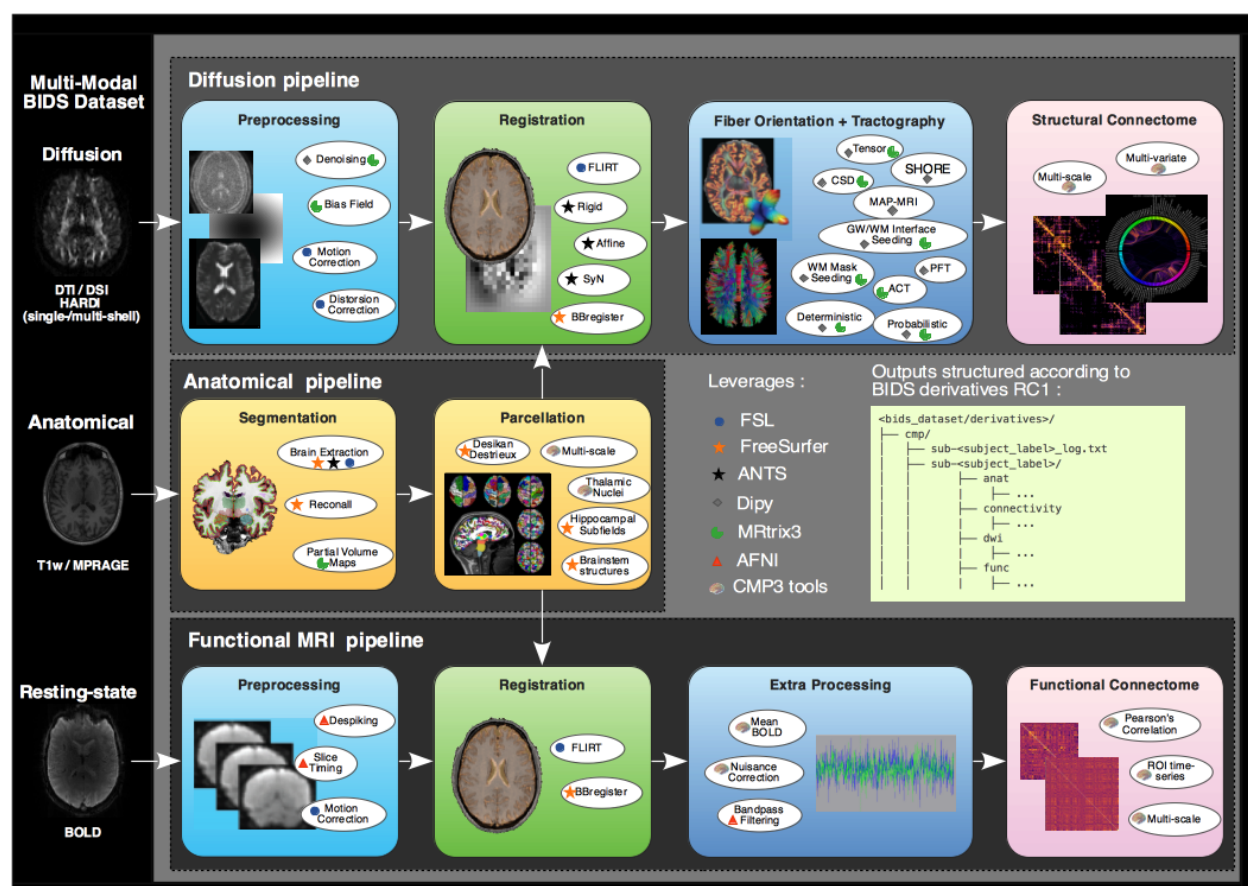
GETTING STARTED

1	About	3
2	License information	5
3	Aknowledgment	7
3.1	Help/Questions	7
4	Eager to contribute?	9
5	Contents	11
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	19
5.5	Outputs of Connectome Mapper 3	75
5.6	Packages and modules	82
5.7	Adopting Datalad for collaboration	222
5.8	Running on a cluster (HPC)	228
5.9	BSD 3-Clause License	230
5.10	Changes	231
5.11	Citing	240
5.12	Contributors	241
5.13	Contributing to Connectome Mapper 3	241
5.14	Support, Bugs and New Feature Requests	245
6	Funding	247
	Bibliography	249
	Python Module Index	251
	Index	253

Latest released version: v3.0.0

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

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



Connectome Mapper 3 is an open-source Python3 image processing pipeline software, with a Graphical User Interface (GUI), that implements full anatomical, diffusion and resting-state MRI processing pipelines, from raw Diffusion / T1 / T2 / BOLD data to multi-resolution connection matrices based on a new version of the Lausanne parcellation atlas, aka Lausanne2018.

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

Portability, reproducibility and replicability are achieved through the distribution of a BIDSApp, a software container image which (1) takes datasets organized following the Brain Imaging Data Structure (BIDS) standard, and which (2) provides a frozen computing environment where versions of all external softwares and libraries are fixed. Accessibility

has been improved to a greater extent by providing an interactive GUI which supports the user in the steps involved in the configuration and execution of the containerized pipelines.

This tool allows you to easily do the following:

- Handle T1 / Diffusion / resting-state MRI data, organized following the Brain Imaging Data Structure (BIDS) standard, 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 with Nipype, which provides a significant speed-up from typical linear, manual processing.
- Easily configure the pipelines and control the execution and outputs of the processing with a GUI.

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

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

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

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

```
$ sudo groupadd docker
```

- Add the current user to the docker group:

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

- Reboot

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

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

Installation

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

- Open a terminal
- Get the latest release (v3.0.0) of the BIDS App:

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

```
$ docker images
```

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

- You are ready to use the Connectome Mapper 3 BIDS App from the terminal. See its [commandline usage](#).

5.1.2 The Connectome Mapper 3 BIDSApp Manager (GUI)

Prerequisites

- Install miniconda3 (Python 3) from <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:

```
$ cd connectomemapper3
$ git fetch
$ git checkout tags/v3.0.0 -b v3.0.0
```

Note: If a few bugs related to the Graphical User Interface were fixed after releasing the version, you might want to use the code at its latest version on the master branch (i.e. `git checkout master`).

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

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

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

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

- 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 `pip`:

```
(py37cmp-gui)$ cd connectomemapper3/
(py37cmp-gui)$ pip install .
```

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

In the future

If you wish to update Connectome Mapper 3 and the Connectome Mapper 3 BIDS App Manager, this could be easily done by (1) updating the git repository to a new tag with `git fetch` and `git checkout tags/|release| -b |release|` and (2) running `pip install ..`

Help/Questions

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

5.2 Connectome Mapper 3 and the BIDS standard

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

This means CMP3 can be executed in two different ways:

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

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

5.2.1 Example BIDS dataset

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

```
ds-example/

README
CHANGES
participants.tsv
dataset_description.json

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

...

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

(continues on next page)

(continued from previous page)

```

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

```

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.

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

5.3.1 Commandline Arguments

The command to run CMP3 follows the [BIDS-Apps](#) definition standard with additional options for loading pipeline configuration files.

Entrypoint script of the BIDS-App Connectome Mapper version v3.0.0

```

usage: connectomemapper3 [-h]
                        [--participant_label PARTICIPANT_LABEL [PARTICIPANT_LABEL ...]]
                        [--session_label SESSION_LABEL [SESSION_LABEL ...]]
                        [--anat_pipeline_config ANAT_PIPELINE_CONFIG]
                        [--dwi_pipeline_config DWI_PIPELINE_CONFIG]
                        [--func_pipeline_config FUNC_PIPELINE_CONFIG]
                        [--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]

```

(continues on next page)

(continued from previous page)

```
[-v]
bids_dir output_dir {participant,group}
```

Positional Arguments

bids_dir	The directory with the input dataset formatted according to the BIDS standard.
output_dir	The directory where the output files should be stored. If you are running group level analysis this folder should be prepopulated with the results of the participant level analysis.
analysis_level	Possible choices: participant, group Level of the analysis that will be performed. Multiple participant level analyses can be run independently (in parallel) using the same output_dir.

Named Arguments

--participant_label	The label(s) of the participant(s) that should be analyzed. The label corresponds to sub-<participant_label> from the BIDS spec (so it does not include “sub-“). If this parameter is not provided all subjects should be analyzed. Multiple participants can be specified with a space separated list.
--session_label	The label(s) of the session that should be analyzed. The label corresponds to ses-<session_label> from the BIDS spec (so it does not include “ses-“). If this parameter is not provided all sessions should be analyzed. Multiple sessions can be specified with a space separated list.
--anat_pipeline_config	Configuration .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 (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 <code>—number_of_threads</code> option flag)
--fs_license	Freesurfer license.txt
--coverage	Run connectomemapper3 with coverage Default: False

--notrack	Do not send event to Google analytics to report BIDS App execution, which is enabled by default. Default: False
-v, --version	show program's version number and exit

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

5.3.2 Participant Level Analysis

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

With the wrappers

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

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

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

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

With the Docker / Singularity Engine

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

Docker

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

```
$ docker run -t --rm -u $(id -u):$(id -g) \
    -v /home/user/data/ds001:/bids_dir -v /home/user/data/ds001/
↳ derivatives:/output_dir (-v /usr/local/freesurfer/license.txt:/bids_dir/
↳ code/license.txt )
    sebastientourbier/connectomemapper-bidsapp:v3.0.0 /bids_dir /output_
↳ dir participant --participant_label 01 (--session_label 01 )
    --anat_pipeline_config /bids_dir/code/ref_anatomical_config.json )
    (--dwi_pipeline_config /bids_dir/code/ref_diffusion_config.json )
    (--func_pipeline_config /bids_dir/code/ref_fMRI_config.json )
    (--number_of_participants_processed_in_parallel 1)
```

Singularity

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

```
$ singularity run --containall --bind /home/user/data/ds001:/bids_dir
↳ --bind /home/user/data/ds001/derivatives:/output_dir --bind /
↳ /usr/local/freesurfer/license.txt:/bids_dir/code/license.txt library://
↳ connectomicslab/default/connectomemapper-bidsapp:v3.0.0 /bids_dir /output_
↳ dir participant --participant_label 01 (--session_label 01 )
    --anat_pipeline_config /bids_dir/code/ref_anatomical_config.json )
    (--dwi_pipeline_config /bids_dir/code/ref_diffusion_config.json )
    (--func_pipeline_config /bids_dir/code/ref_fMRI_config.json )
    (--number_of_participants_processed_in_parallel 1)
```

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

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

Note: At least a configuration file describing the processing stages of the anatomical pipeline should be provided. Diffusion and/or Functional MRI pipeline are performed only if a configuration file is set. The generation of such configuration files, the execution of the BIDS App docker image and output 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](#). 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 py37cmp-gui
```

or:

```
$ conda activate py37cmp-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
```

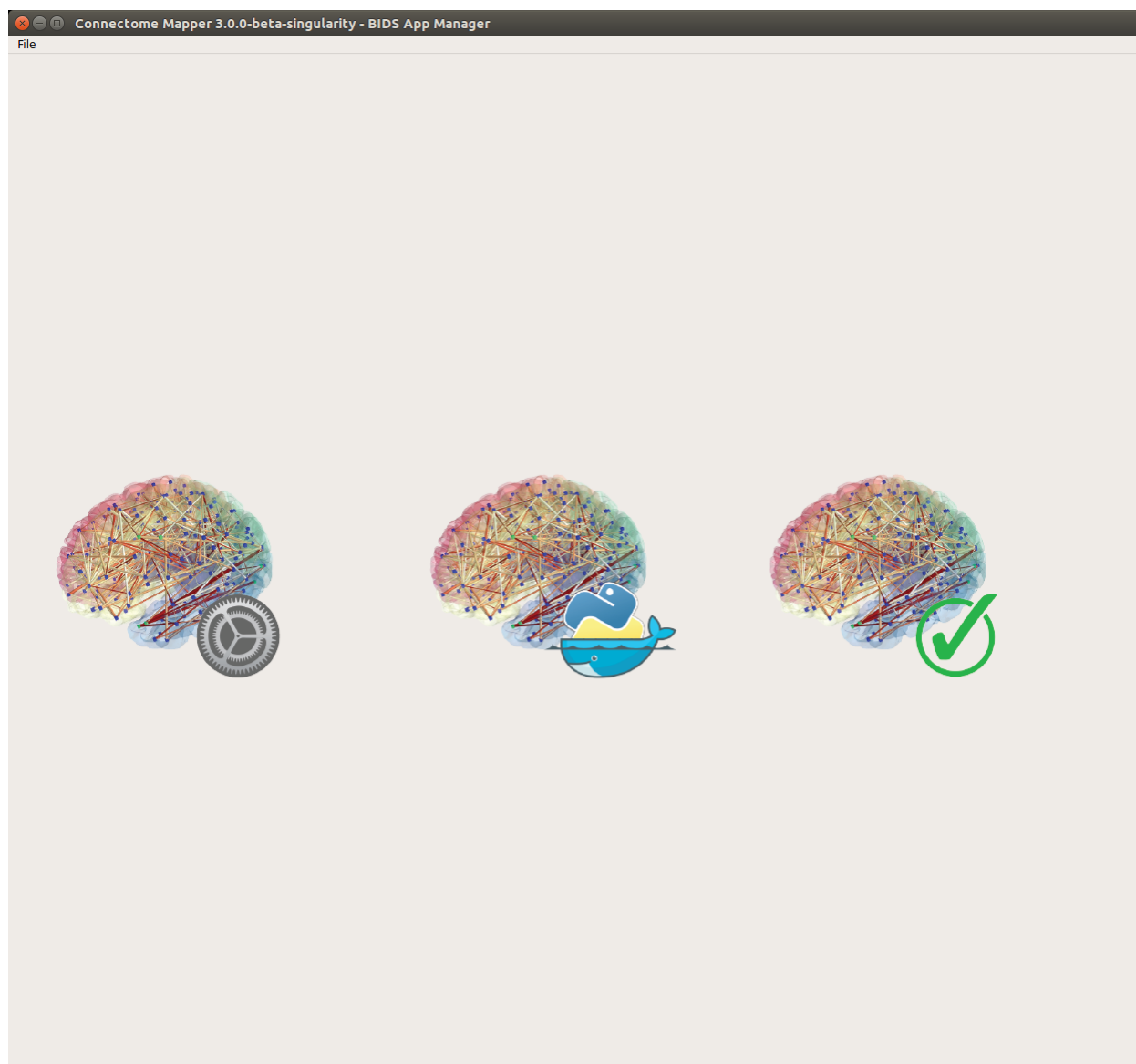


Fig. 1: Main window of the Connectome Mapper BIDS App Manager

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”.

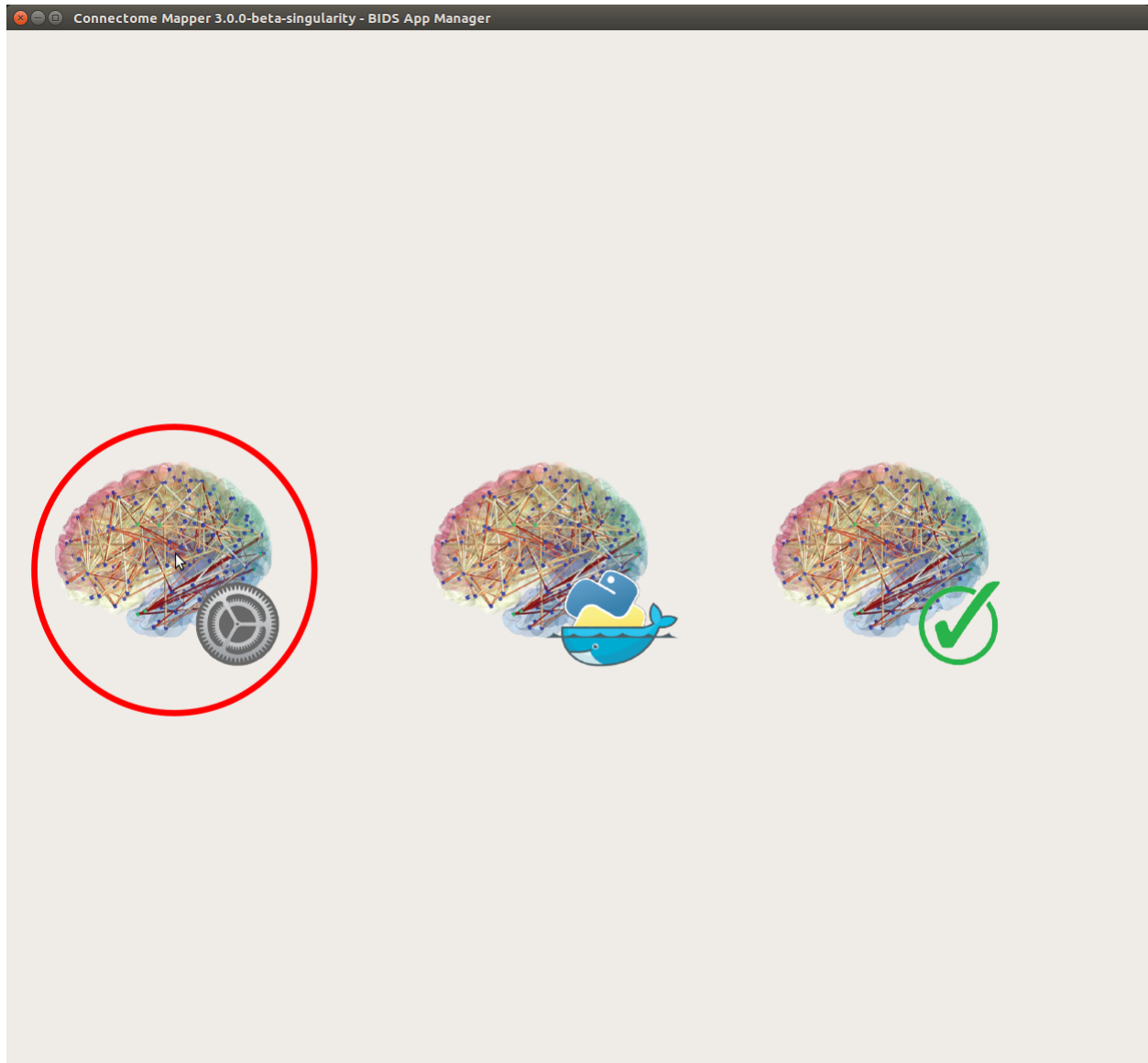
If ssh connection is used, make sure to enable the “install via ssh” and to provide all connection details (IP address / Remote host name, remote user, remote password)

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](#).

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.

Anatomical pipeline stages

Segmentation

Performs tissue segmentation using Freesurfer.

Freesurfer

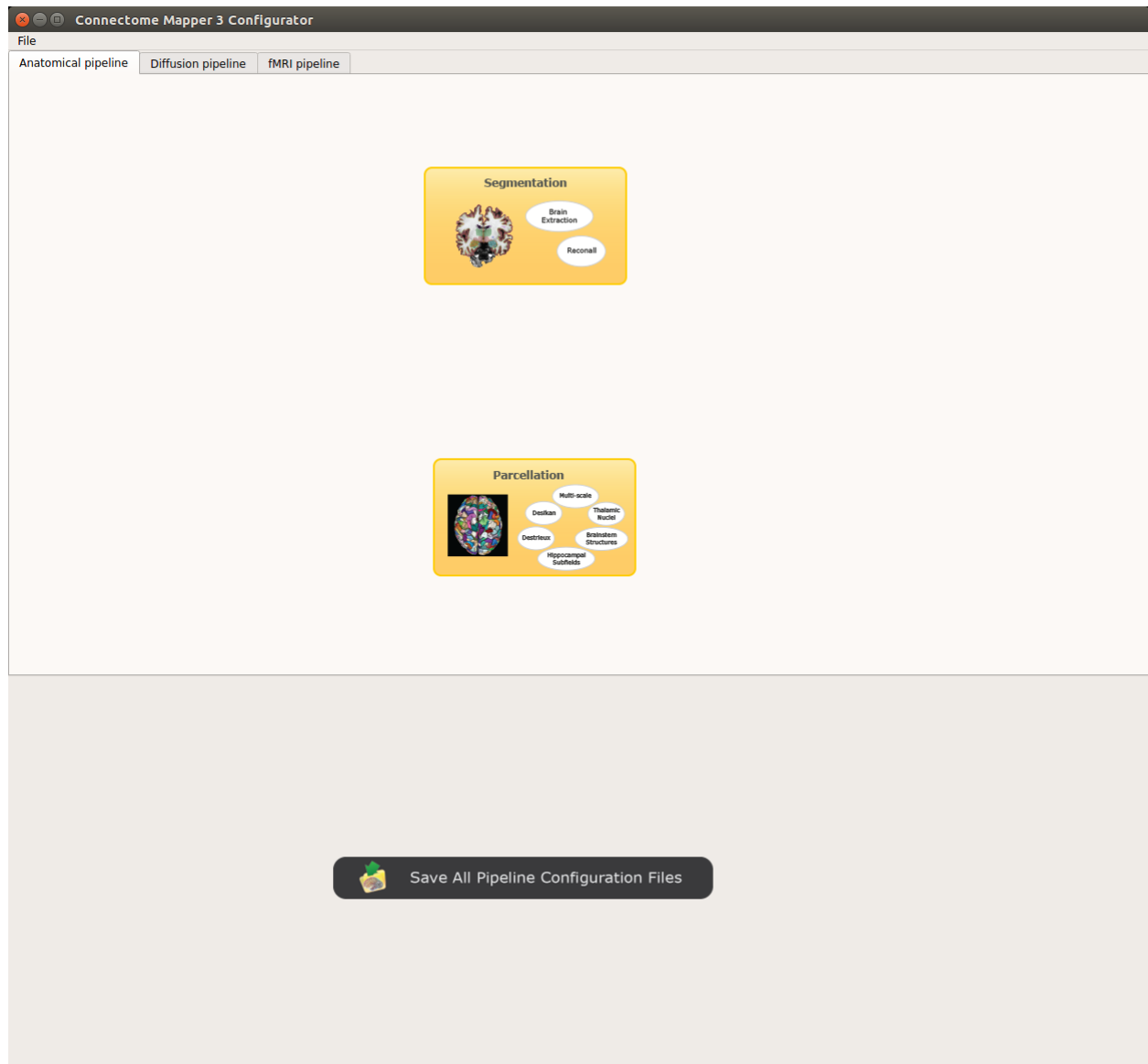


Fig. 2: Configurator Window of the Connectome Mapper

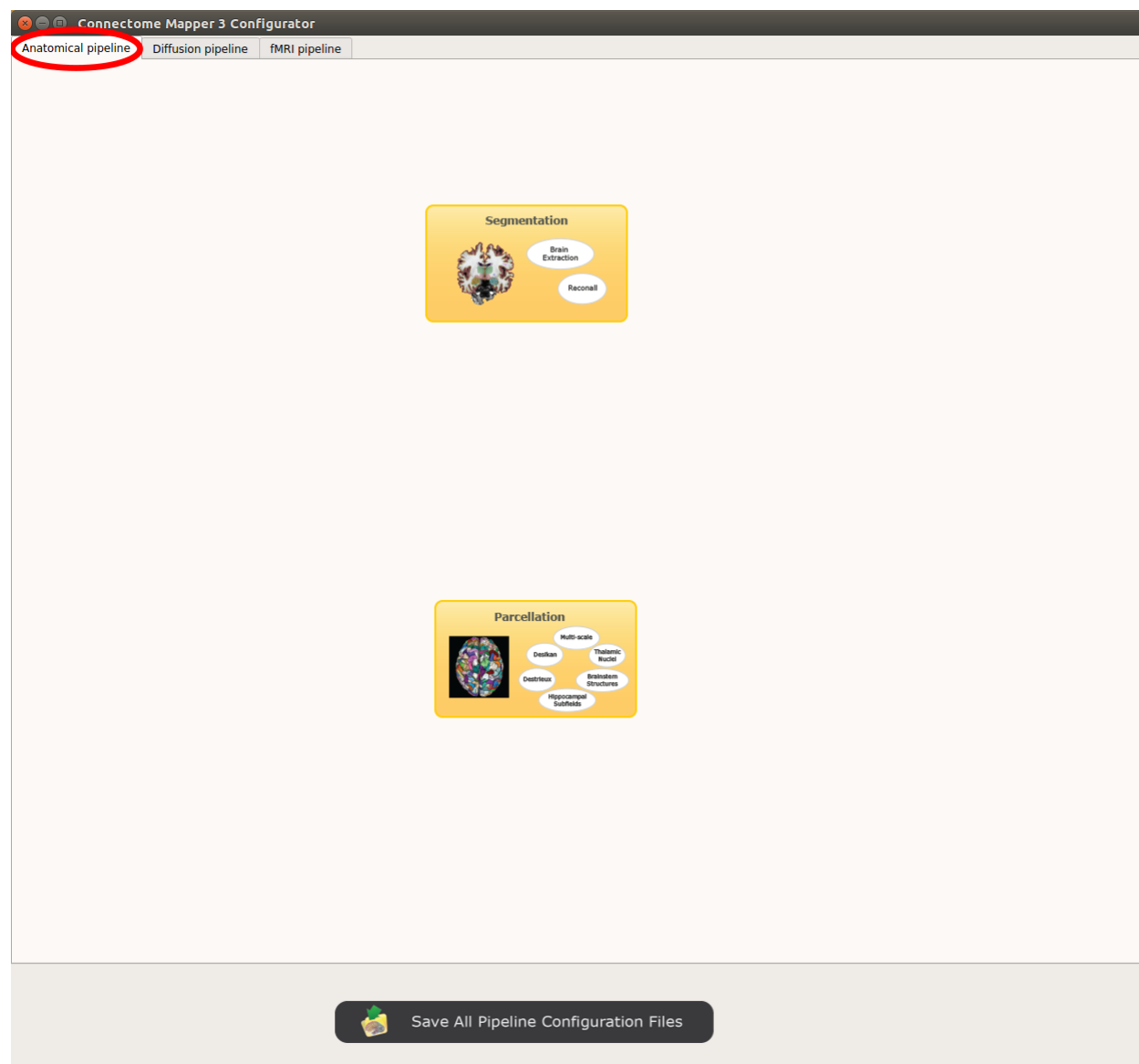
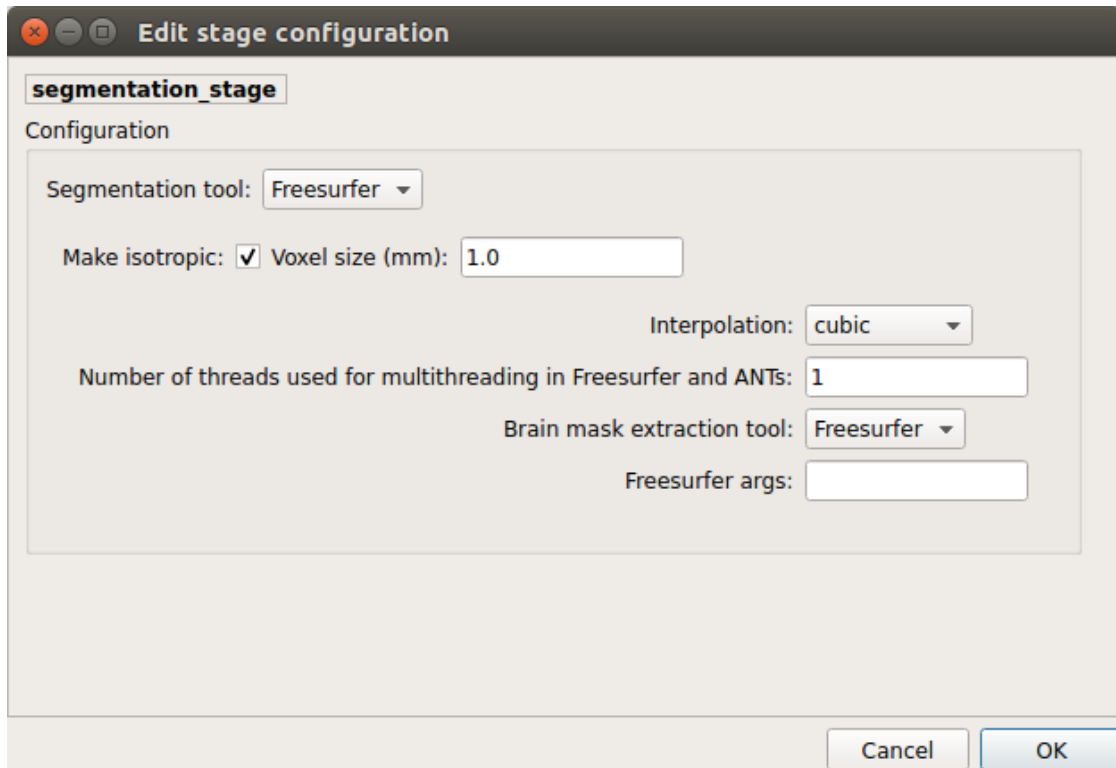


Fig. 3: Panel for configuration of anatomical pipeline stages



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

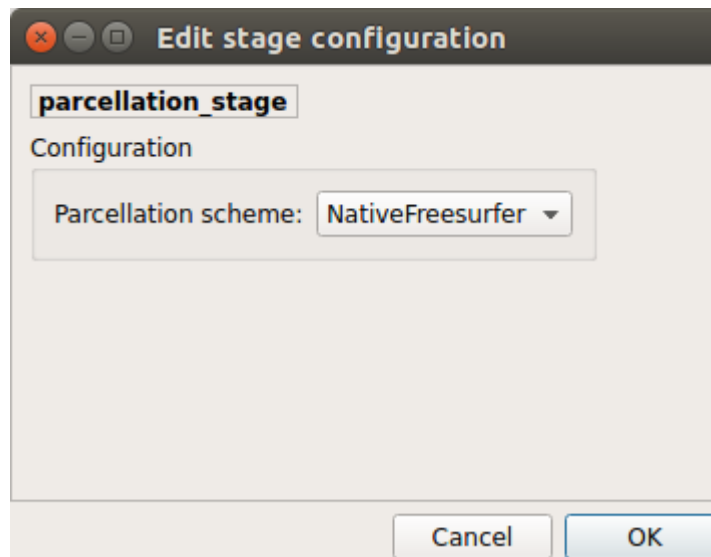
Note: If you have already Freesurfer output data available, CMP3 can use them if there are placed in your output / derivatives directory.

Parcellation

Generates the Native Freesurfer or Lausanne2008/Lausanne2018 parcellation from Freesurfer data.

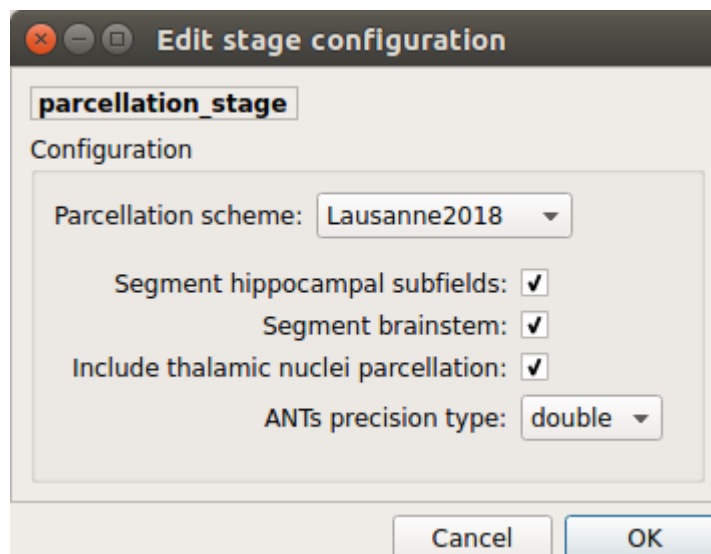
Parcellation scheme

- *NativeFreesurfer:*



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

- *Lausanne2018*:



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

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

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

Diffusion pipeline stages

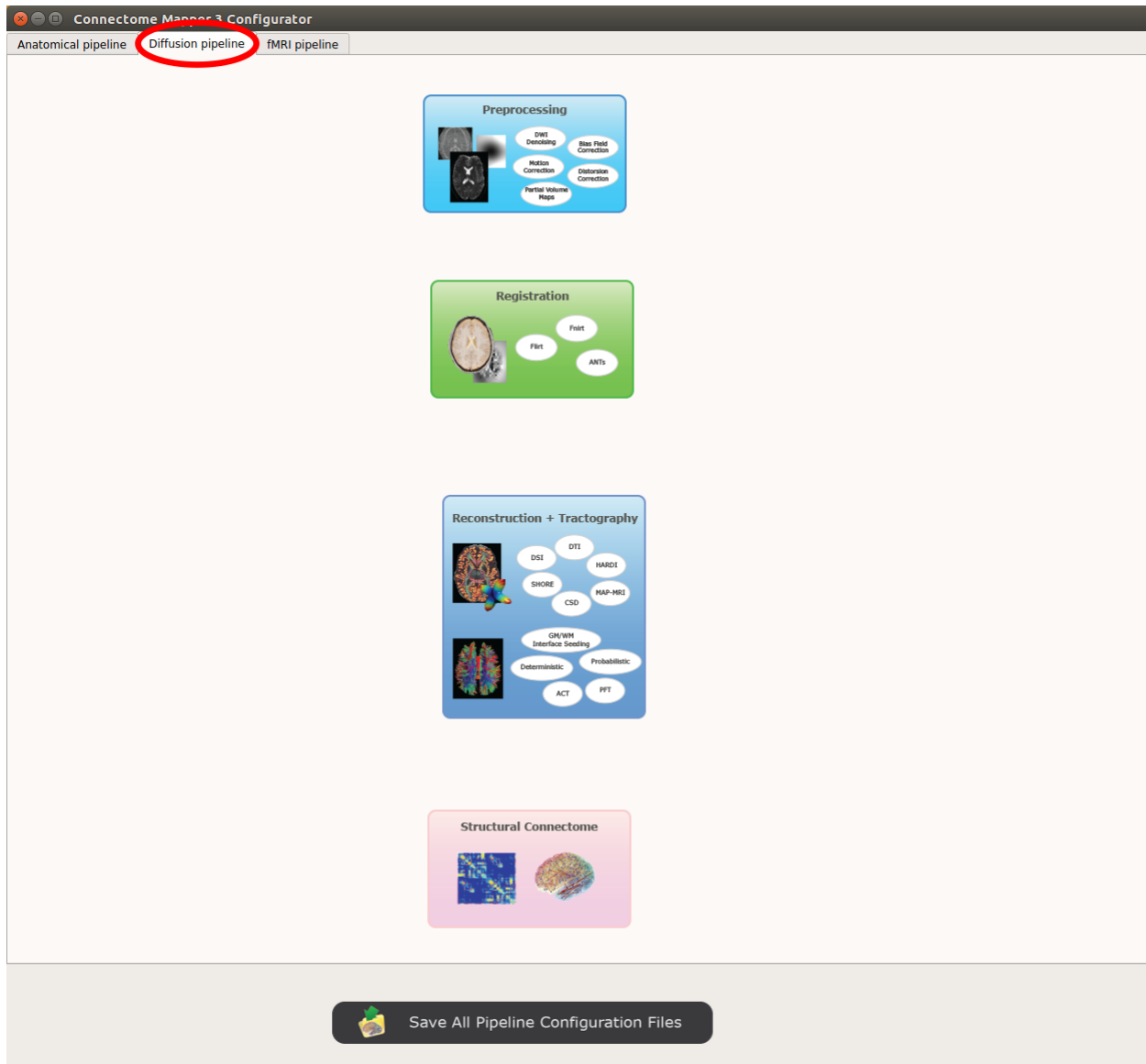


Fig. 4: Panel for configuration of diffusion pipeline stages

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

Eddy current correction

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

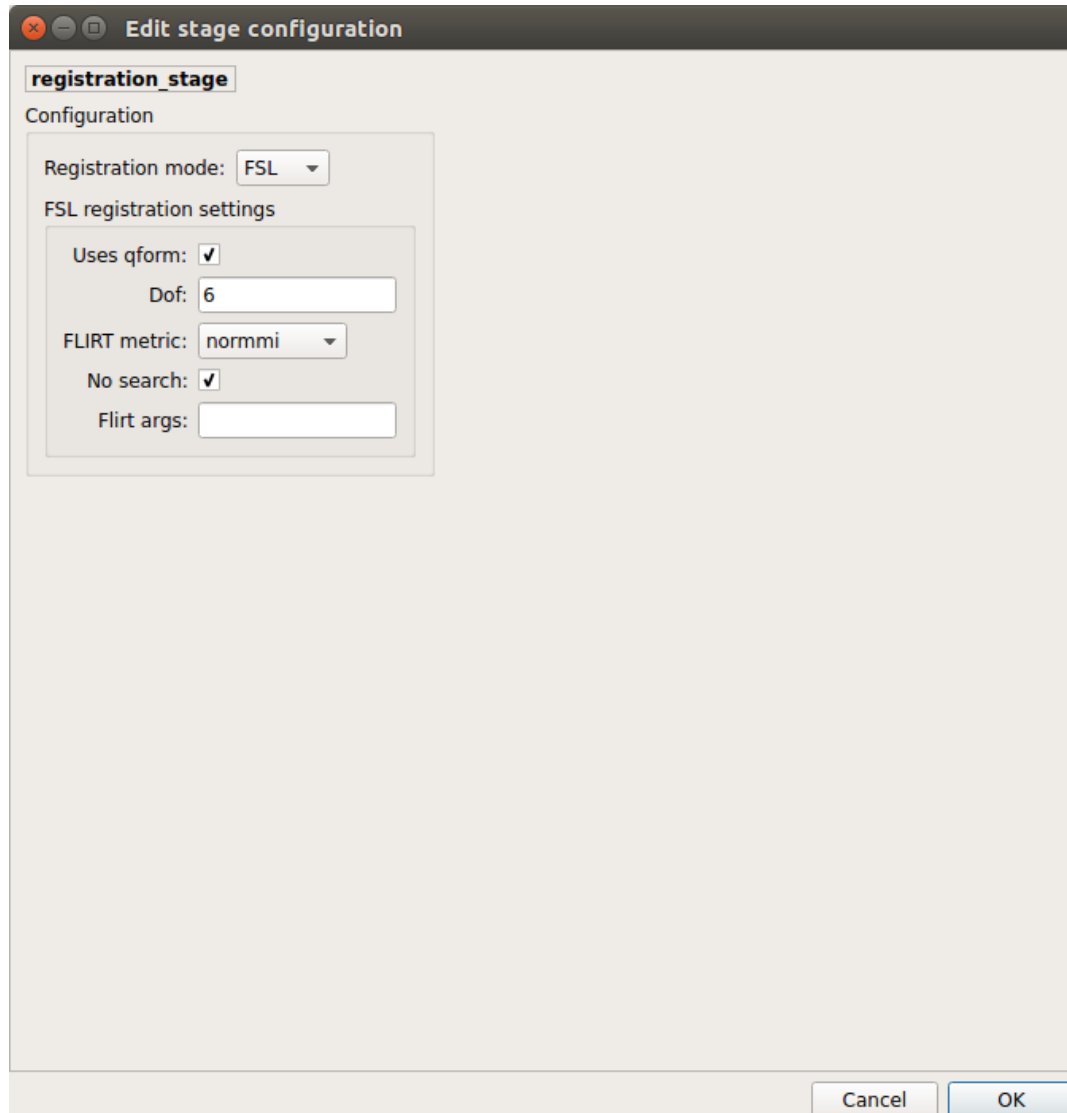
Resampling

Resample morphological and diffusion data to F0 x F1 x F2 mm³

Registration

Registration mode

- FSL (Linear):



The screenshot shows a window titled "Edit stage configuration" with a tab labeled "registration_stage". Inside the window, under the "Configuration" section, there are several settings for FSL registration:

- Registration mode:** A dropdown menu set to "FSL".
- FSL registration settings:** A sub-section containing:
 - Uses qform:** A checkbox that is checked.
 - Dof:** A text input field containing the value "6".
 - FLIRT metric:** A dropdown menu set to "normmi".
 - No search:** A checkbox that is checked.
 - Flirt args:** An empty text input field.

At the bottom right of the dialog, there are "Cancel" and "OK" buttons.

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

- 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 diffusion b=0

Diffusion reconstruction and tractography

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

Reconstruction tool

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

- SHORE:

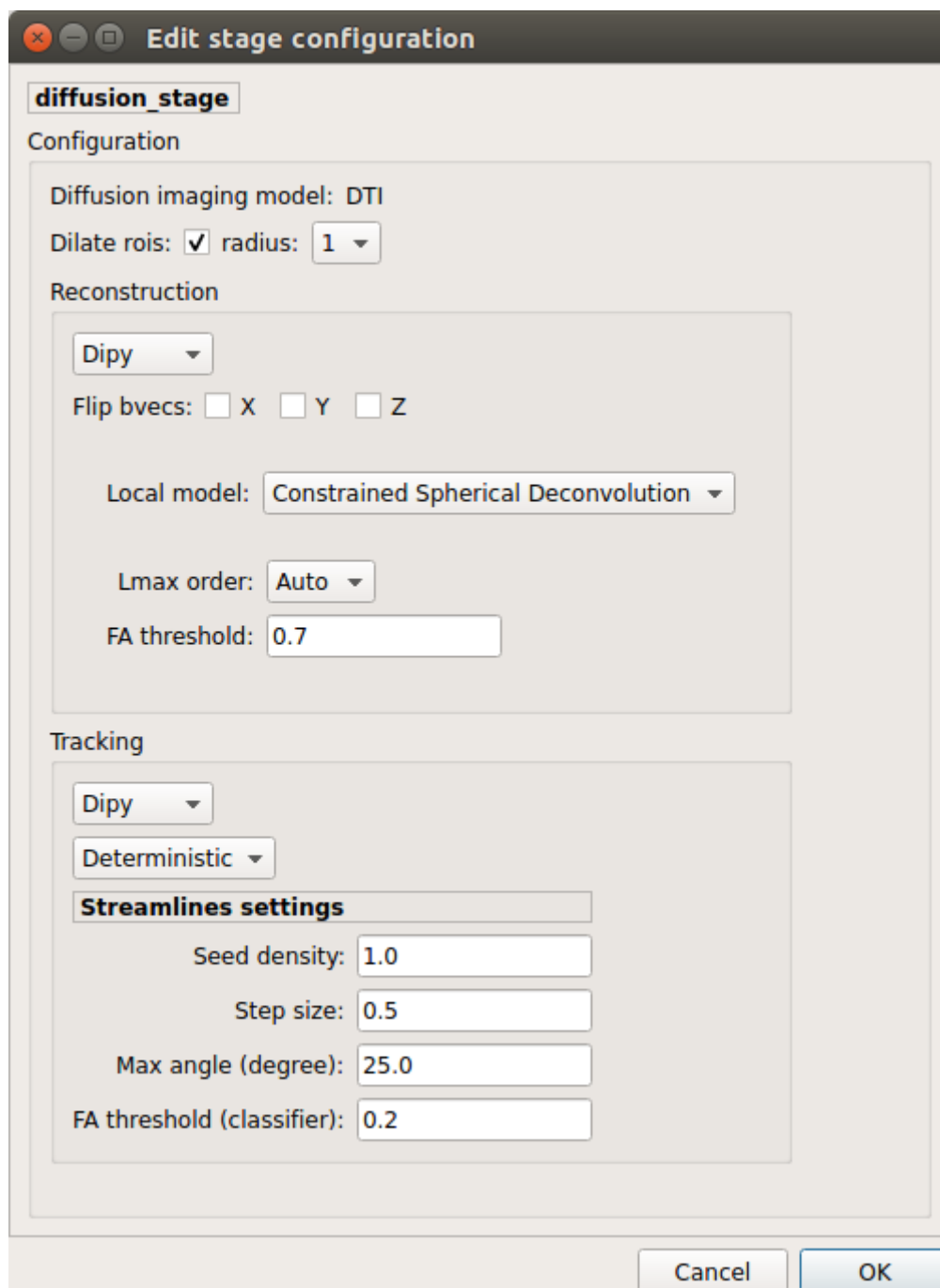


Fig. 5: Diffusion stage configuration window

The screenshot shows the 'Reconstruction' panel with the following settings:

- Method: Dipy (dropdown)
- Flip bvecs: ☐ X ☐ Y ☐ Z
- Parameters of SHORE reconstruction model (grouped box):
 - Radial order: 6 (dropdown)
 - Scale factor (zeta): 700 (text input)
 - Radial regularization constant: 1e-08 (text input)
 - Angular regularization constant: 1e-08 (text input)
 - Diffusion time (s): 0.0253302959106 (text input)
 - Constrain the optimization such that $E(0) = 1$: ☐
 - Constrain the propagator to be positive: ☐
- Mapmri: ☐

SHORE performed only on DSI data

- Tensor:

The screenshot shows the 'Reconstruction' panel with the following settings:

- Method: Dipy (dropdown)
- Flip bvecs: ☐ X ☐ Y ☐ Z
- Local model: Tensor (dropdown)

Tensor performed only on DTI data

- CSD:

The screenshot shows the 'Reconstruction' panel with the following settings:

- Method: Dipy (dropdown)
- Flip bvecs: ☐ X ☐ Y ☐ Z
- Local model: Constrained Spherical Deconvolution (dropdown)
- Lmax order: Auto (dropdown)
- FA threshold: 0.7 (text input)

CSD performed on DTI and multi-shell data

- MAP_MRI:

Reconstruction

Dipy ▾

Flip bvecs: ☐ X ☐ Y ☐ Z

Local model: Constrained Spherical Deconvolution ▾

Lmax order: Auto ▾

FA threshold: 0.7

Mapmri: ☒

MAP_MRI settings

Radial order: 8

Small delta: 0.02 Big delta: 0.5

Laplacian regularization: ☒ Laplacian weighting: 0.05

Positivity constraint: ☒

MAP-MRI performed only on multi-shell data

MRtrix: perform CSD reconstruction.

- CSD:

Reconstruction

MRtrix ▾

Flip gradient table: ☐ X ☐ Y ☐ Z

Local model: Constrained Spherical Deconvolution ▾

Lmax order: Auto ▾

Normalize to b0: ☐

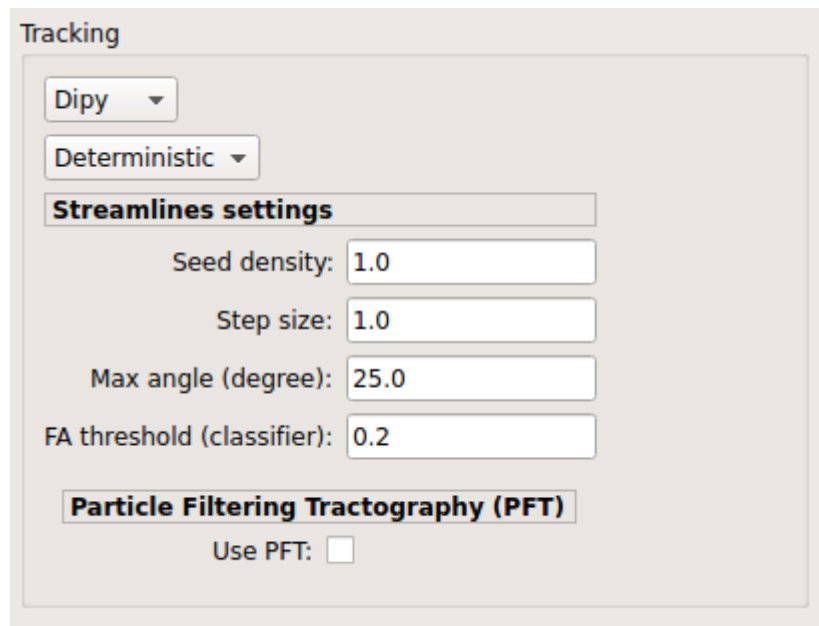
FA threshold: 0.7

CSD performed on DTI and multi-shell data

Tractography tool

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

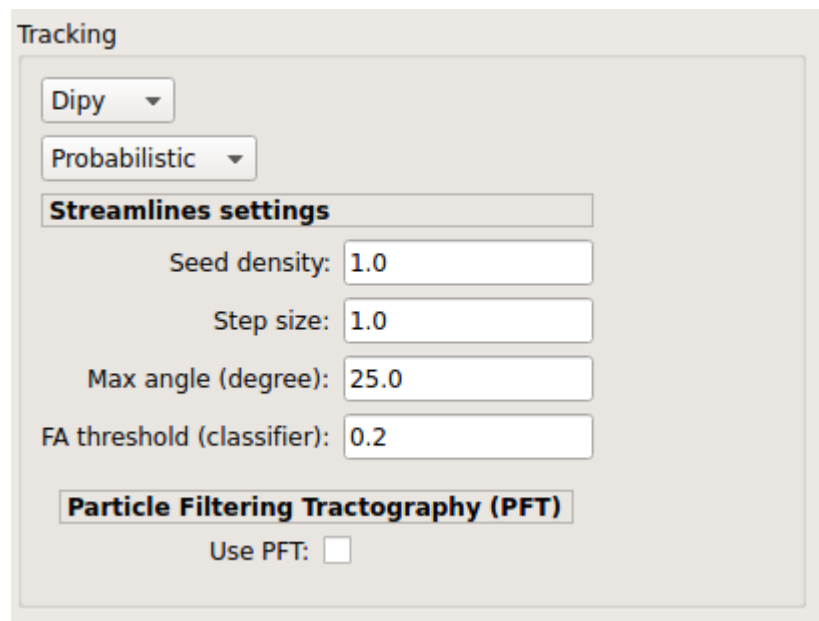
- Deterministic tractography:



The image shows a 'Tracking' settings panel. At the top, there is a dropdown menu set to 'Dipy'. Below it is another dropdown menu set to 'Deterministic'. A section titled 'Streamlines settings' contains four input fields: 'Seed density' with a value of 1.0, 'Step size' with a value of 1.0, 'Max angle (degree)' with a value of 25.0, and 'FA threshold (classifier)' with a value of 0.2. At the bottom, there is a section titled 'Particle Filtering Tractography (PFT)' with a checkbox labeled 'Use PFT:' which is currently unchecked.

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

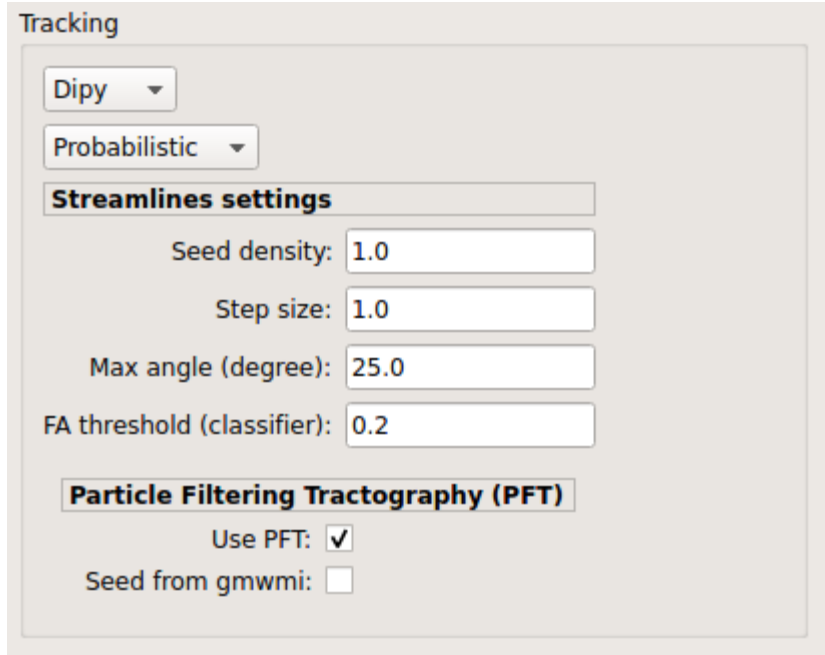
- Probabilistic tractography:



The image shows a 'Tracking' settings panel, similar to the one above but with the second dropdown menu set to 'Probabilistic'. The 'Streamlines settings' section remains the same with values: Seed density: 1.0, Step size: 1.0, Max angle (degree): 25.0, and FA threshold (classifier): 0.2. The 'Particle Filtering Tractography (PFT)' section at the bottom also remains the same with the 'Use PFT:' checkbox unchecked.

Probabilistic tractography (iFOD2) performed on SHORE or CSD reconstruction

- Probabilistic particle filtering tractography (PFT):



The screenshot shows a 'Tracking' window with the following settings:

- Method: Dipy (dropdown)
- Tracking type: Probabilistic (dropdown)
- Streamlines settings** (grouped box):
 - Seed density: 1.0 (text input)
 - Step size: 1.0 (text input)
 - Max angle (degree): 25.0 (text input)
 - FA threshold (classifier): 0.2 (text input)
- Particle Filtering Tractography (PFT)** (grouped box):
 - Use PFT: ☒ (checkbox)
 - Seed from gmwmi: ☐ (checkbox)

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:

Tracking

MRtrix ▾

Deterministic ▾

Streamline settings

Desired number of tracks: 5000000

Min length: 4.0 Max length: 200.0

Angle: 45.0

Curvature radius: 0.0

Step size: 0.5

Cutoff value: 0.05

Anatomically-Constrained Tractography (ACT)

Use ACT: ☐

Streamline filtering

Filter tractogram with SIFT: ☐

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

- Deterministic anatomically-constrained tractography (ACT):

Tracking

MRtrix ▾

Deterministic ▾

Streamline settings

Desired number of tracks: 5000000

Min length: 4.0 Max length: 200.0

Angle: 45.0

Curvature radius: 0.0

Step size: 0.5

Cutoff value: 0.05

Anatomically-Constrained Tractography (ACT)

Use ACT: ☒

Crop at gmwmi: ☐

Backtrack: ☐

Seed from gmwmi: ☐

Streamline filtering

Filter tractogram with SIFT: ☐

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:

Tracking

MRtrix ▾

Probabilistic ▾

Streamline settings

Desired number of tracks: 5000000

Min length: 4.0 Max length: 200.0

Angle: 45.0

Curvature radius: 1.0

Step size: 0.5

Cutoff value: 0.05

Anatomically-Constrained Tractography (ACT)

Use ACT: ☐

Streamline filtering

Filter tractogram with SIFT: ☐

Probabilistic tractography (iFOD2) performed on SHORE or CSD reconstruction

- Probabilistic anatomically-constrained tractography (ACT):

Tracking

MRtrix ▾

Probabilistic ▾

Streamline settings

Desired number of tracks: 5000000

Min length: 4.0 Max length: 200.0

Angle: 45.0

Curvature radius: 1.0

Step size: 0.5

Cutoff value: 0.05

Anatomically-Constrained Tractography (ACT)

Use ACT: ☒

Crop at gmwmi: ☐

Backtrack: ☐

Seed from gmwmi: ☐

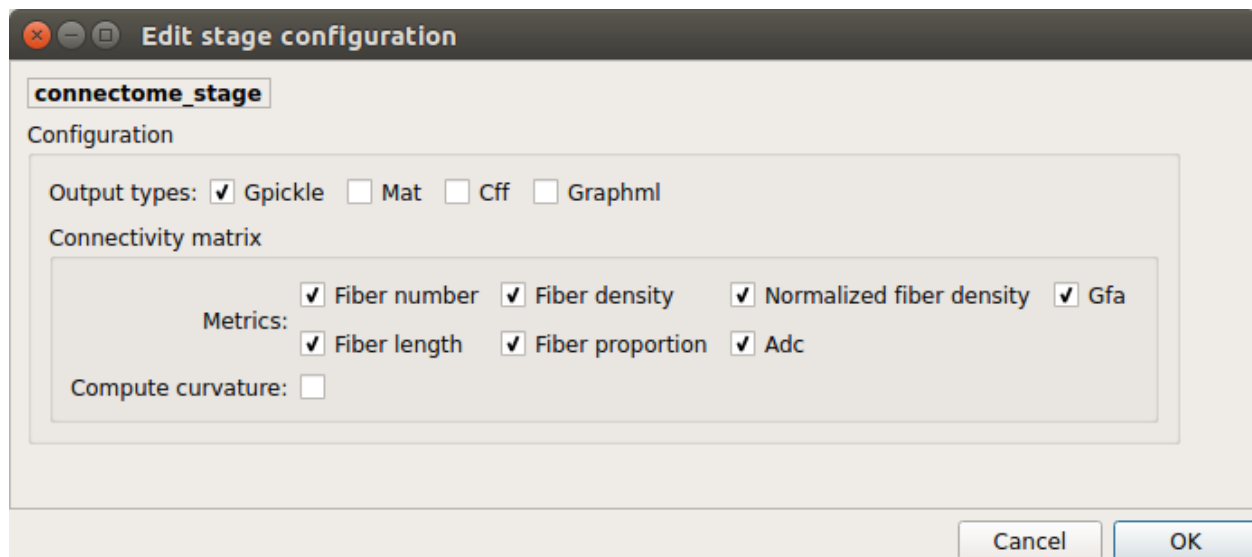
Streamline filtering

Filter tractogram with SIFT: ☐

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

Connectome

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



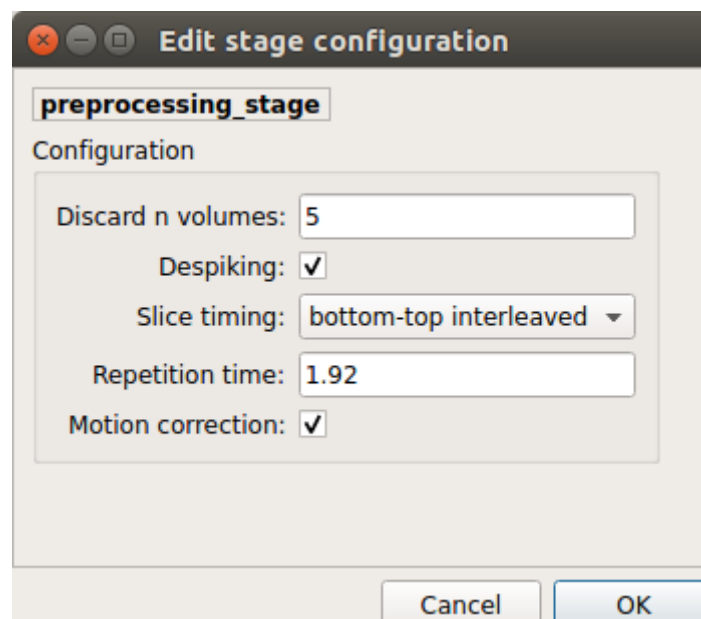
Output types

Select in which formats the connectivity matrices should be saved.

FMRI pipeline stages

Preprocessing

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



Discard n volumes

Discard n volumes from further analysis

Despiking

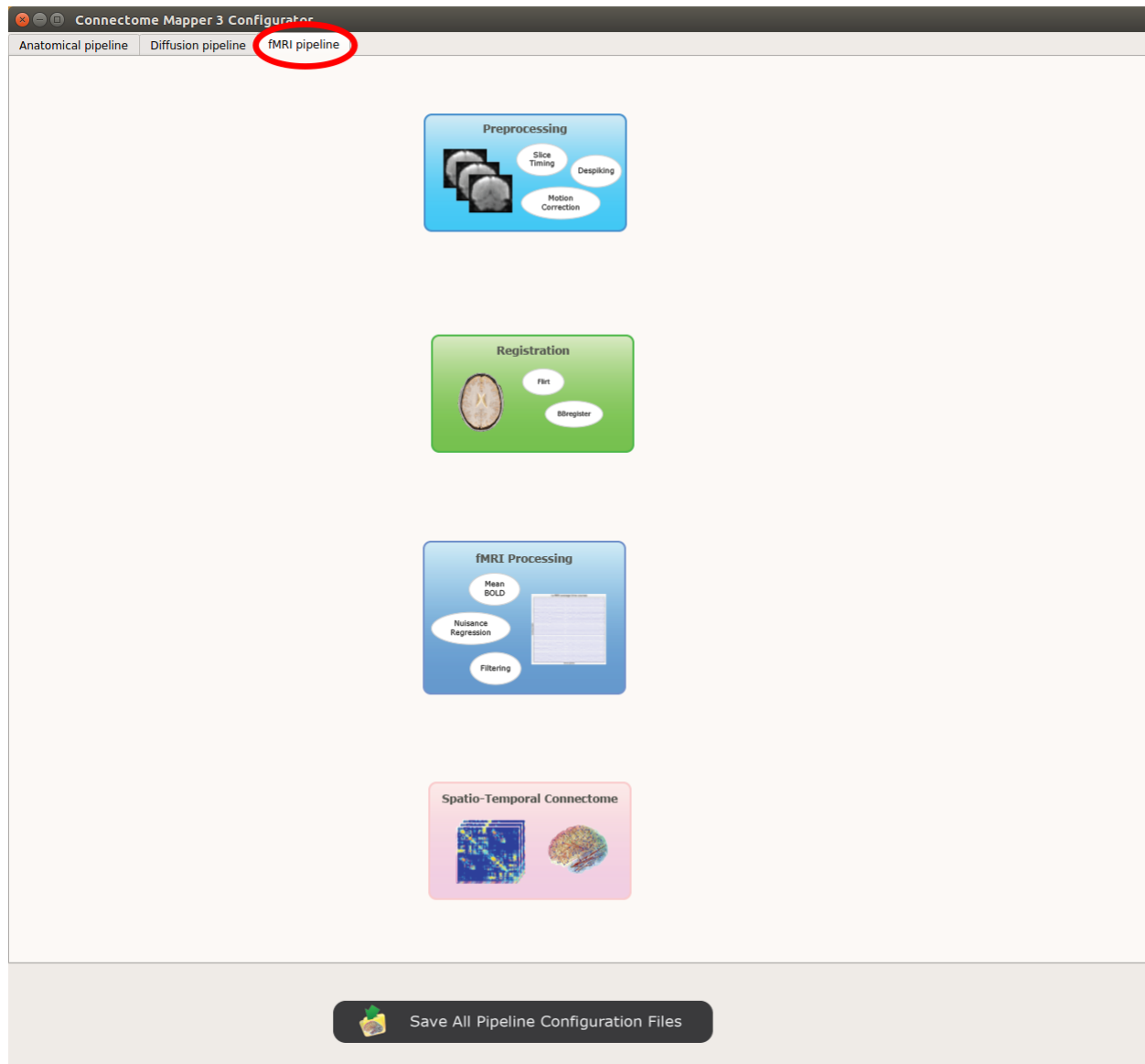


Fig. 6: Panel for configuration of fMRI pipeline stages

Perform despiking of the BOLD signal using AFNI.

Slice timing and Repetition time

Perform slice timing correction using FSL's slicetimer.

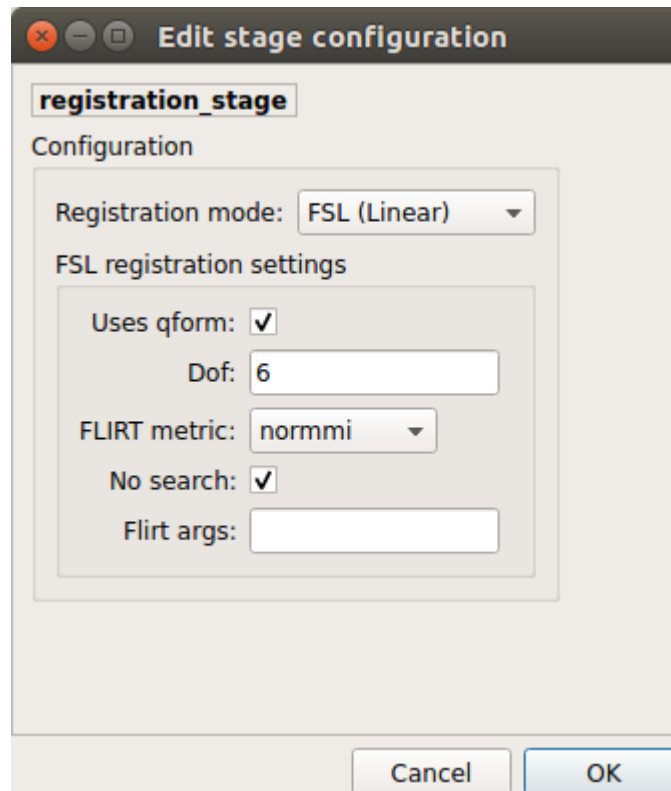
Motion correction

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

Registration

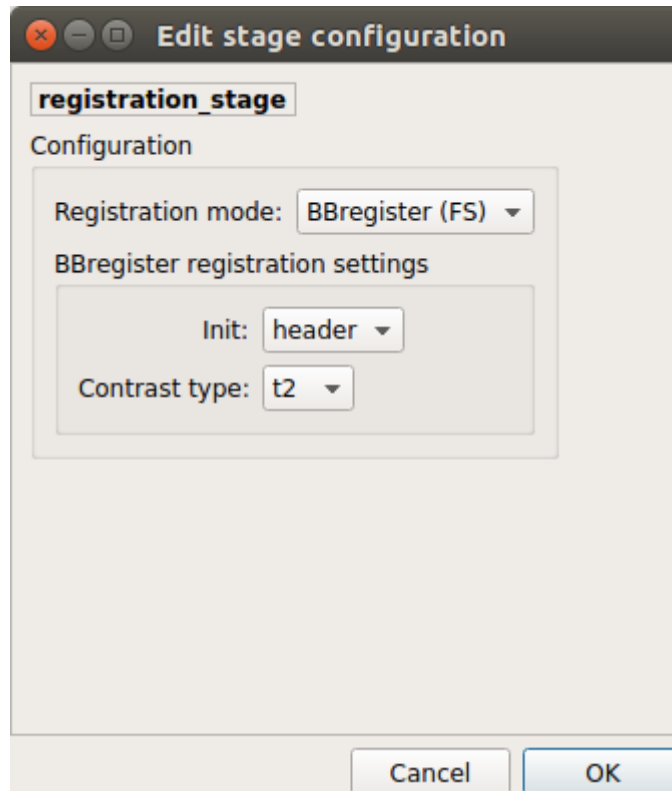
Registration mode

- FSL (Linear):



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

- BBregister (FS)



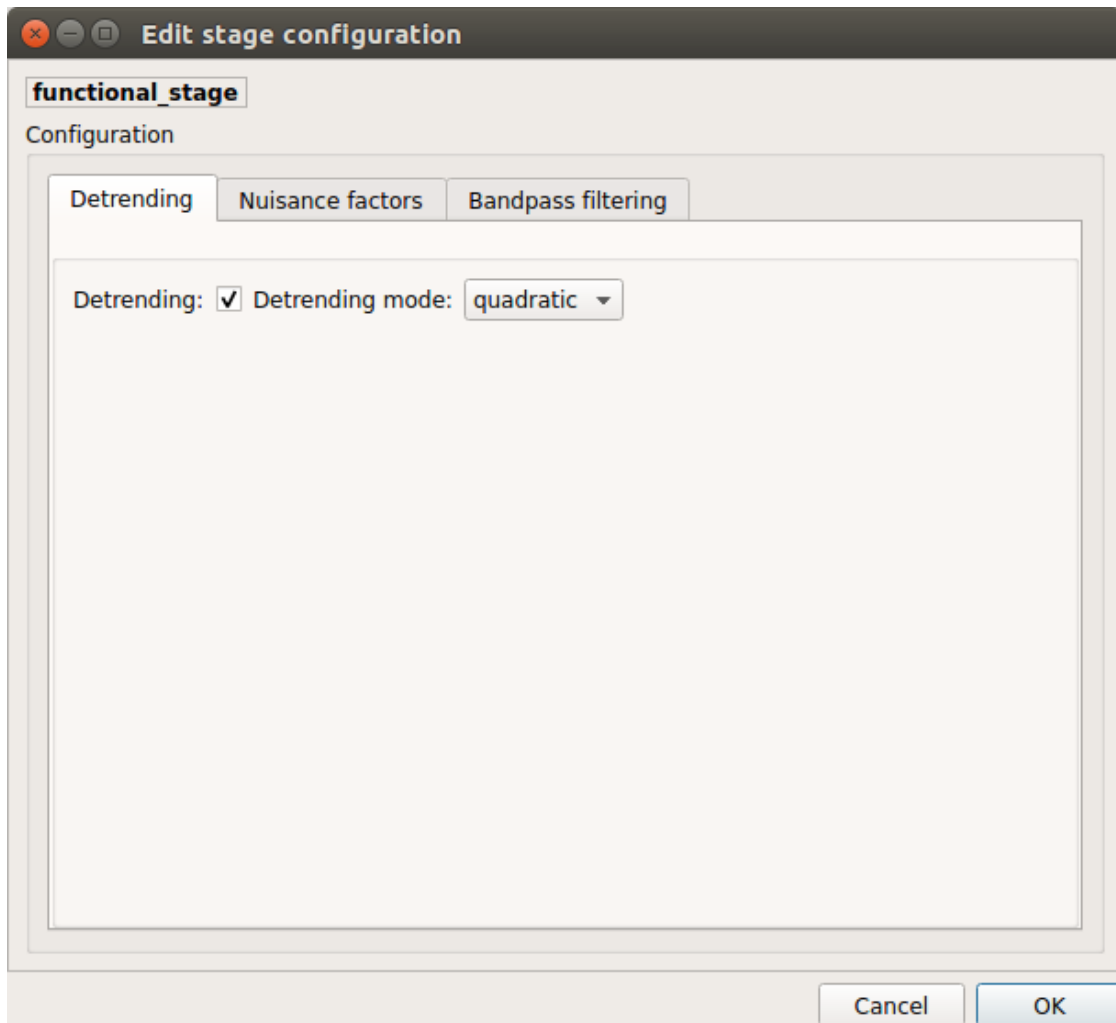
Perform linear registration using Freesurfer BBregister tool from T1 to mean BOLD via T2.

Warning: development in progress

fMRI processing

Performs detrending, nuisance regression, bandpass 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.

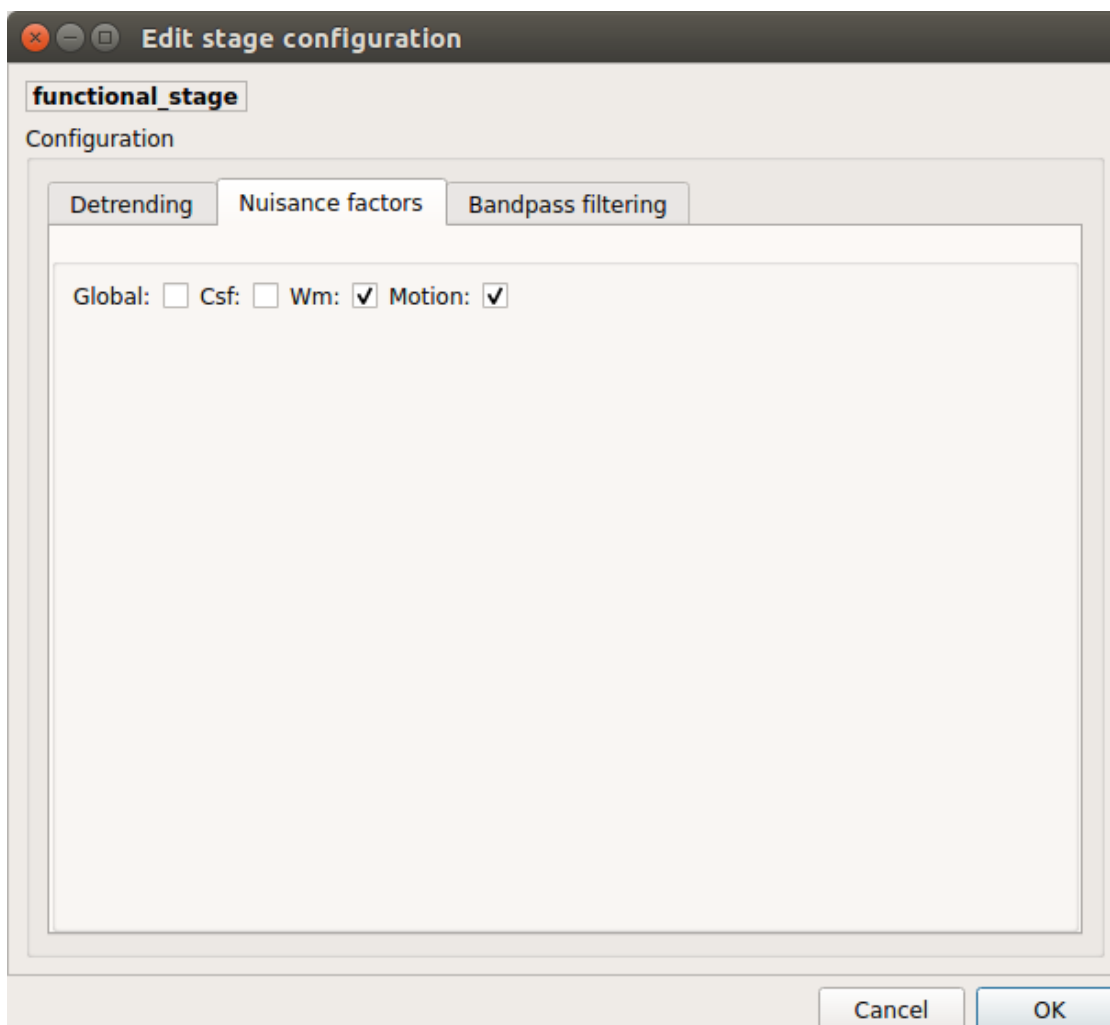
Detrending



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

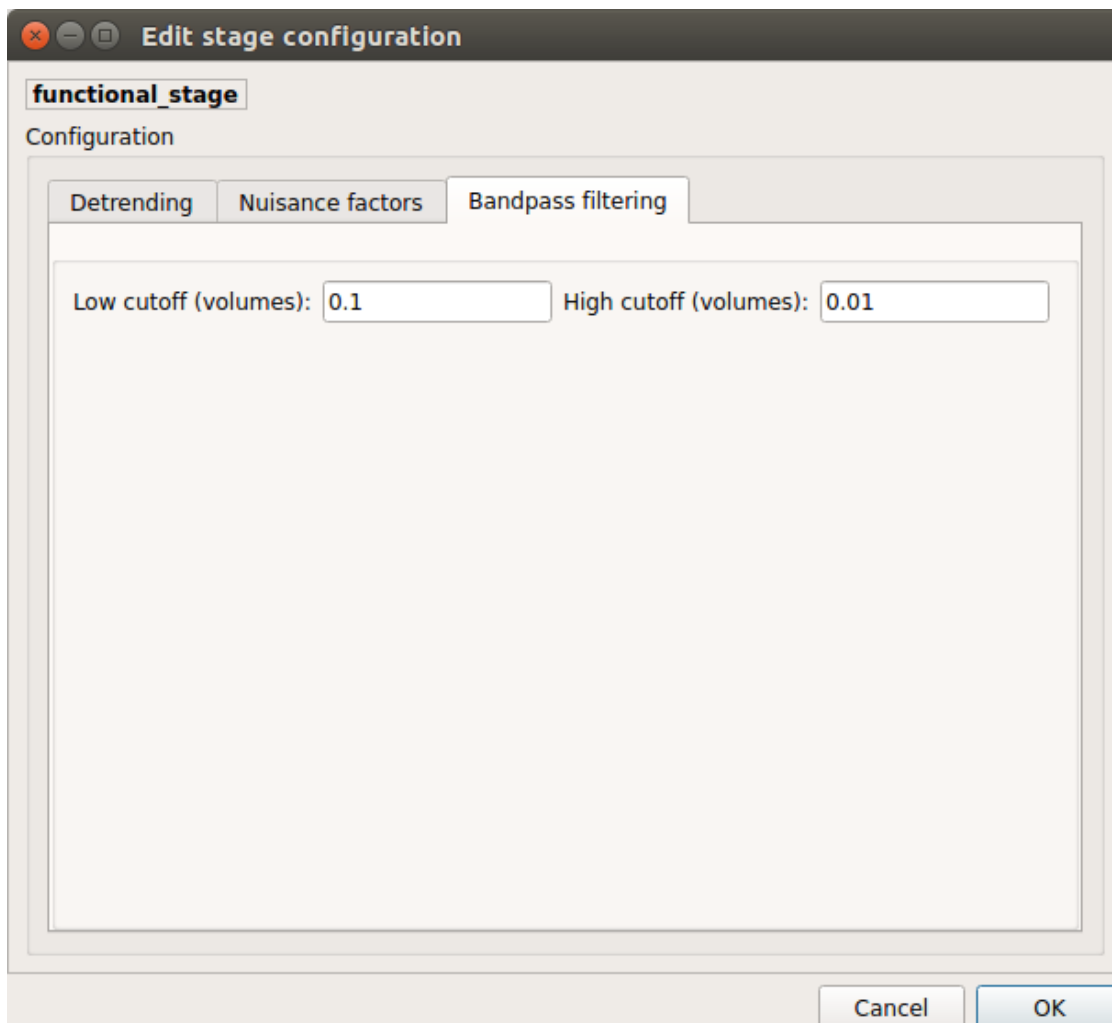
Nuisance regression



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

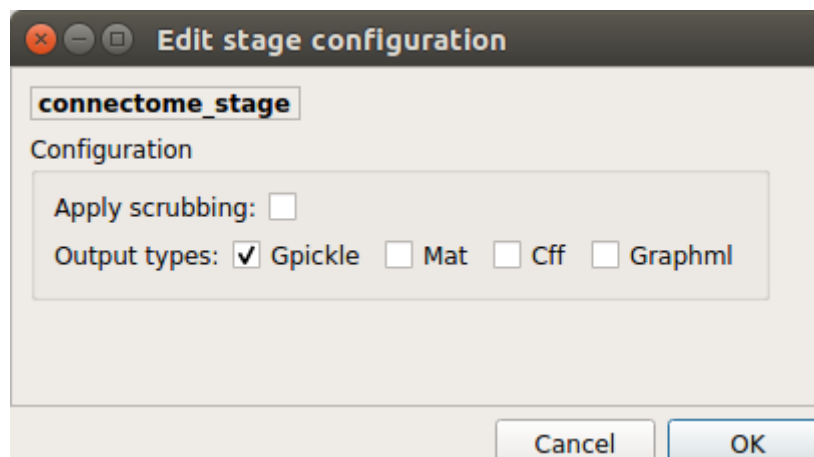
Bandpass filtering



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...)

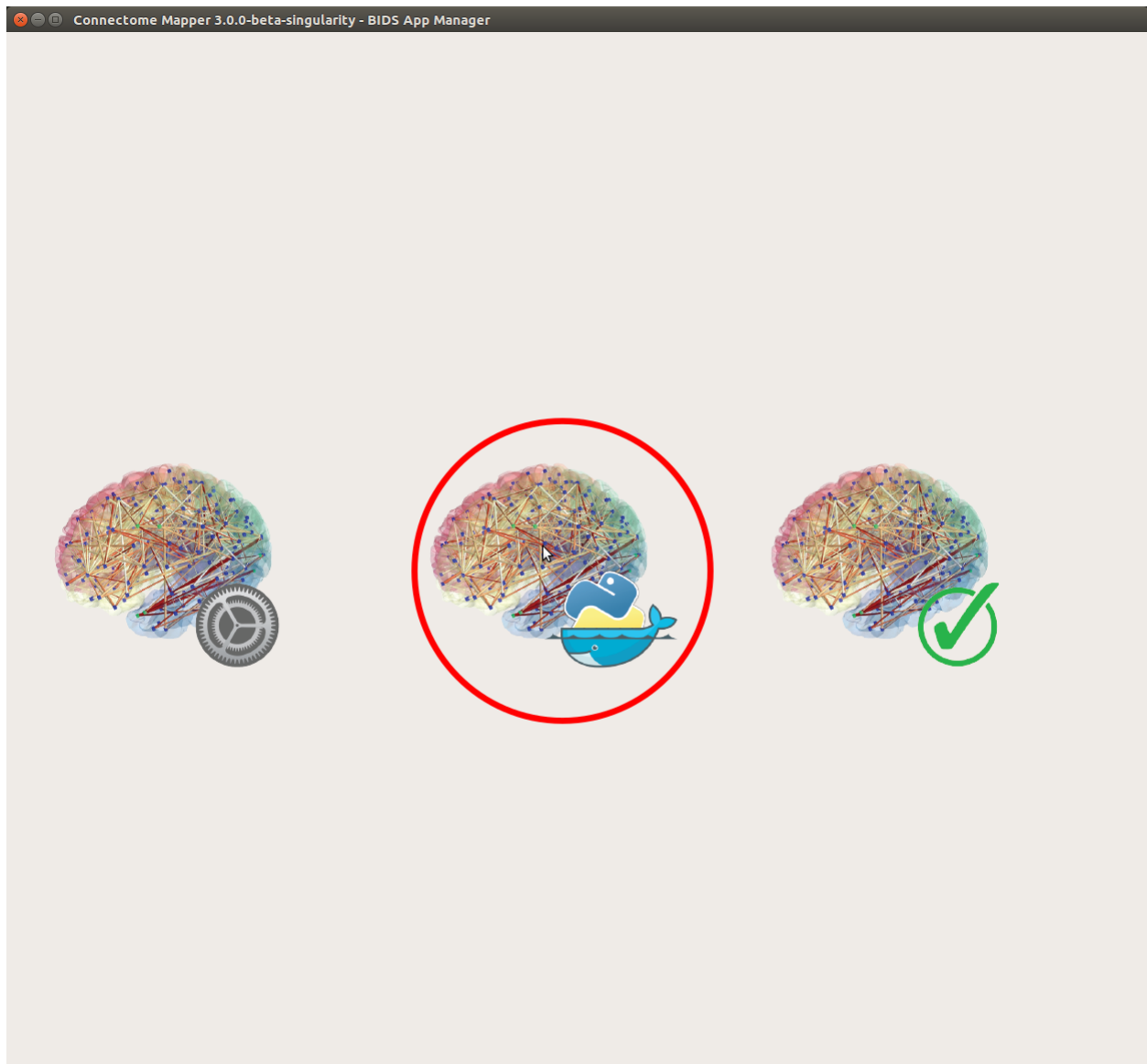
Nipype

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

5.4.5 Run the BIDS App

Start the Connectome Mapper BIDS App GUI

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



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

Run configuration

- Select the output directory for data derivatives

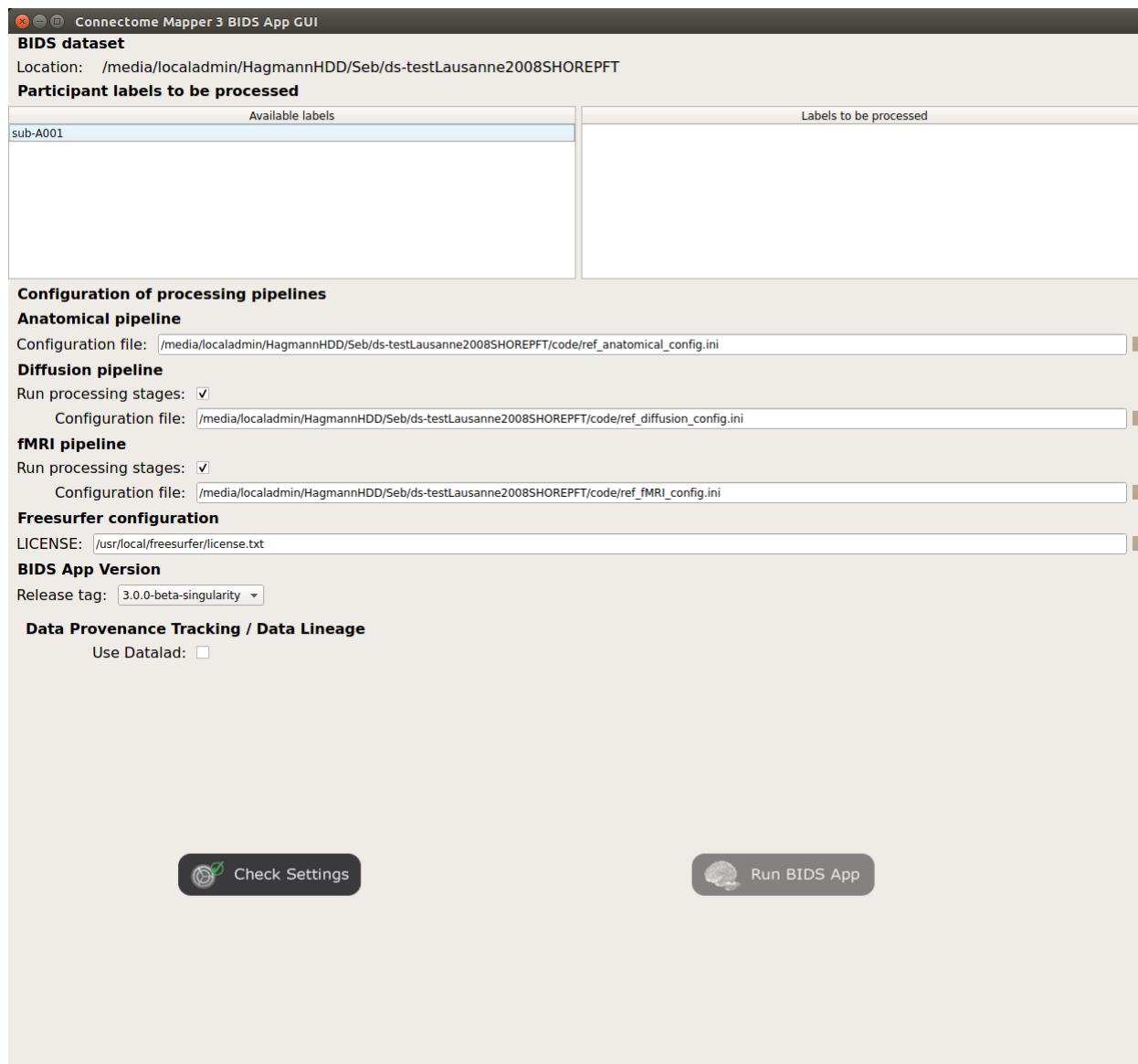


Fig. 7: Window of the Connectome Mapper BIDS App GUI

Connectome Mapper 3 BIDS App GUI

BIDS App Version
Tag: v3.0.0-RC3

BIDS dataset
Input directory: /home/localadmin/Desktop/hcp-retest-d2
Output directory: /home/localadmin/Desktop/hcp-retest-d2/derivatives

Participant labels to be processed

Available labels	Labels to be processed
sub-103818	

Parallel processing Number of participants processed in parallel: 1

Advanced execution settings for each participant process

Multithreading Random number generators
 Number of OpenMP threads: 1 Set seed of ANTS random number generator:
 Set number of threads used by ANTs: Seed: 1234
 Number of ITK threads used by ANTs registration: 1 Set seed of MRtrix random number generator:
 Seed: 1234

Configuration of processing pipelines

Anatomical pipeline
Configuration file: /home/localadmin/Desktop/hcp-retest-d2/code/ref_anatomical_config.ini

Diffusion pipeline
Run processing stages: ☒
Configuration file: /home/localadmin/Desktop/hcp-retest-d2/code/ref_diffusion_config.ini

fMRI pipeline
Run processing stages: ☒
Configuration file: /home/localadmin/Desktop/hcp-retest-d2/code/ref_fmri_config.ini

FreeSurfer configuration
LICENSE: /usr/local/freesurfer/license.txt

Data Provenance Tracking / Data Lineage
Use DataLad: ☐

- Select the subject labels to be processed

Connectome Mapper 3 BIDS App GUI

BIDS App Version
Tag: v3.0.0-RC3

BIDS dataset
Input directory: /home/localadmin/Desktop/hcp-retest-d2
Output directory: /home/localadmin/Desktop/hcp-retest-d2/derivatives

Participant labels to be processed

Available labels	Labels to be processed
sub-103818	sub-103818

Parallel processing Number of participants processed in parallel: 1

Advanced execution settings for each participant process

Multithreading Random number generators
 Number of OpenMP threads: 1 Set seed of ANTS random number generator:
 Set number of threads used by ANTs: Seed: 1234
 Number of ITK threads used by ANTs registration: 1 Set seed of MRtrix random number generator:
 Seed: 1234

Configuration of processing pipelines

Anatomical pipeline
Configuration file: /home/localadmin/Desktop/hcp-retest-d2/code/ref_anatomical_config.ini

Diffusion pipeline
Run processing stages: ☒
Configuration file: /home/localadmin/Desktop/hcp-retest-d2/code/ref_diffusion_config.ini

fMRI pipeline
Run processing stages: ☒
Configuration file: /home/localadmin/Desktop/hcp-retest-d2/code/ref_fmri_config.ini

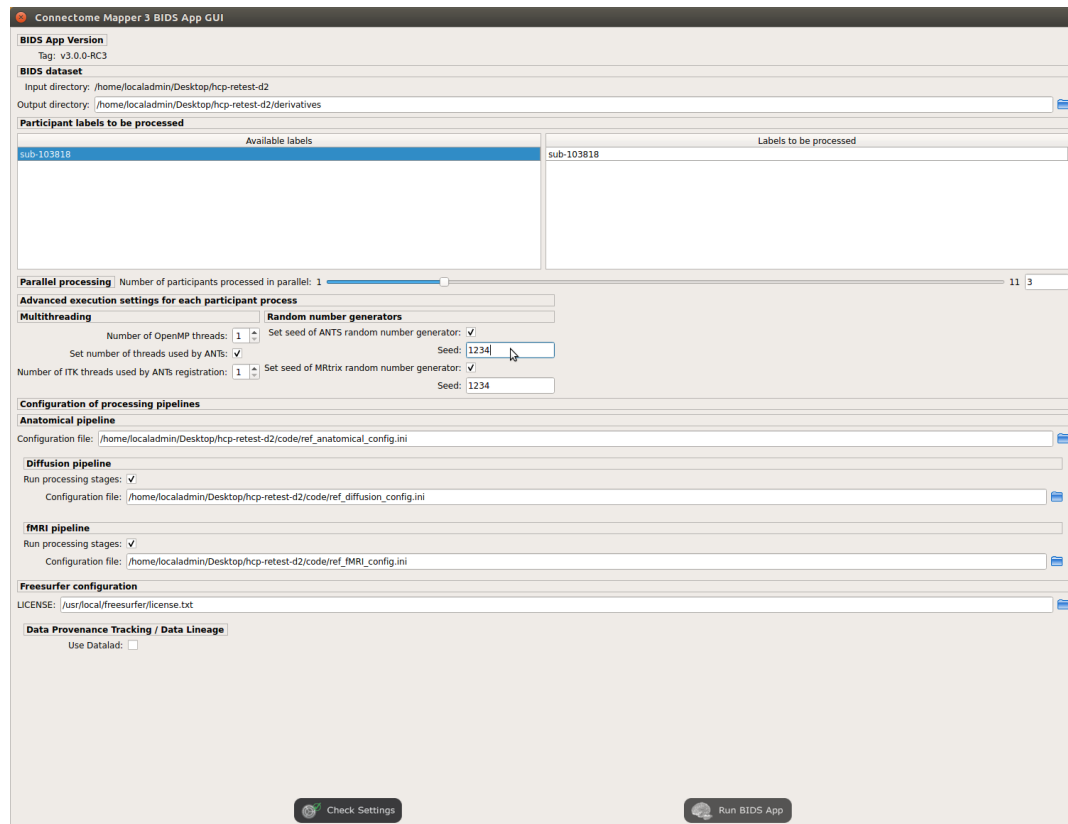
FreeSurfer configuration
LICENSE: /usr/local/freesurfer/license.txt

Data Provenance Tracking / Data Lineage
Use DataLad: ☐

- Tune the number of subjects to be processed in parallel

The screenshot displays the Connectome Mapper 3 BIDS App GUI. At the top, it shows the app version (v3.0.0-RC3) and the BIDS dataset path. Below this, there are fields for the input directory and output directory. A section titled 'Participant labels to be processed' contains two lists: 'Available labels' and 'Labels to be processed', both showing 'sub-103818'. The 'Parallel processing' section features a slider for the number of participants processed in parallel, currently set to 1. The 'Advanced execution settings for each participant process' section includes a 'Multithreading' subsection with a 'Number of OpenMP threads' set to 1, and a 'Random number generators' subsection with 'Set seed of ANTs random number generator' and 'Set seed of MRtrix random number generator' both set to 1234. The 'Configuration of processing pipelines' section includes 'Anatomical pipeline', 'Diffusion pipeline', and 'fMRI pipeline', each with a 'Run processing stages' checkbox and a 'Configuration file' field. The 'Freesurfer configuration' section includes a 'LICENSE' field. At the bottom, there is a 'Data Provenance Tracking / Data Lineage' section with a 'Use DataLad' checkbox. The interface concludes with 'Check Settings' and 'Run BIDS App' buttons.

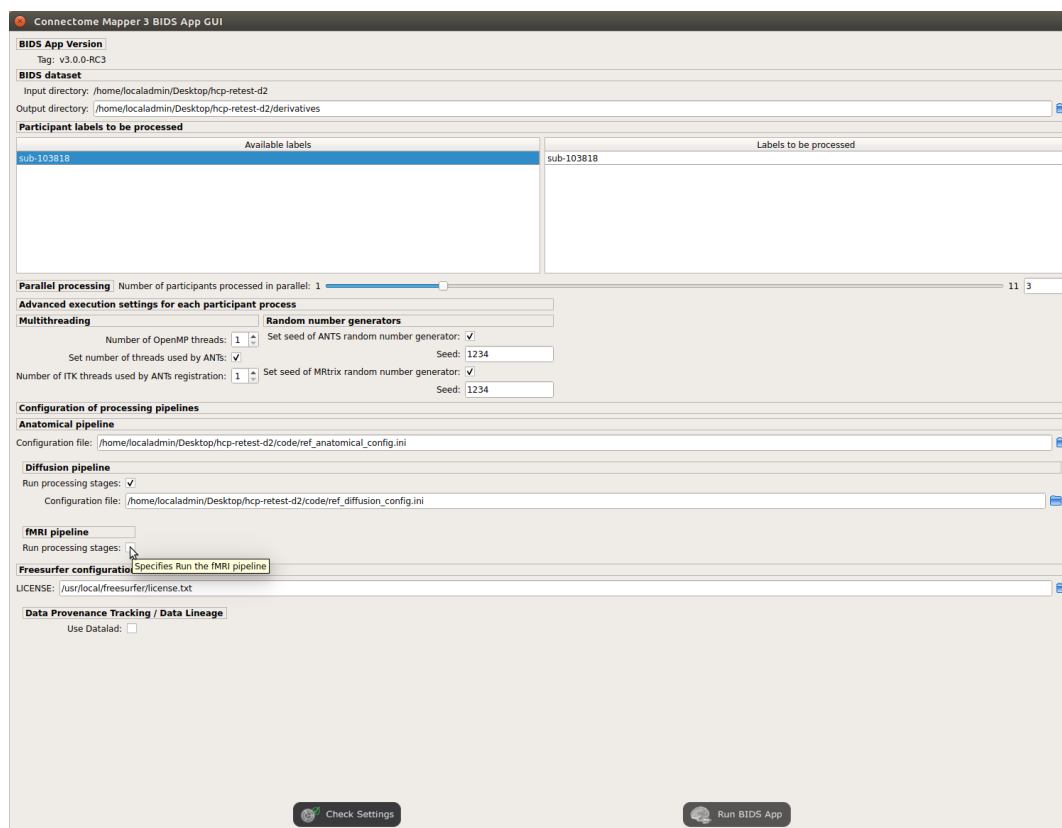
- Tune the advanced execution settings for each subject process. This include finer control on the number of threads used by each process as well as on the seed value of ANTs and MRtrix random number generators.



The screenshot shows the Connectome Mapper 3 BIDS App GUI. At the top, it displays the BIDS App Version (v3.0.0-RC3) and the BIDS dataset path. Below this, there are two lists: 'Available labels' and 'Labels to be processed', both containing the subject ID 'sub-103818'. The 'Parallel processing' section shows a slider for the number of participants processed in parallel, set to 1. The 'Advanced execution settings for each participant process' section includes 'Multithreading' and 'Random number generators' options. Under 'Multithreading', there are checkboxes for 'Number of OpenMP threads' (set to 1), 'Set number of threads used by ANTs' (checked), and 'Number of ITK threads used by ANTs registration' (set to 1). Under 'Random number generators', there are checkboxes for 'Set seed of ANTs random number generator' (checked) and 'Set seed of MRtrix random number generator' (checked), both with a seed value of 1234. The 'Configuration of processing pipelines' section includes 'Anatomical pipeline', 'Diffusion pipeline', and 'fMRI pipeline', each with a 'Run processing stages' checkbox (all checked) and a configuration file path. The 'FreeSurfer configuration' section includes a 'LICENSE' field and a 'Data Provenance Tracking / Data Lineage' section with a 'Use DataLad' checkbox (unchecked). At the bottom, there are two buttons: 'Check Settings' and 'Run BIDS App'.

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

- Check/Uncheck the pipelines to be performed



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

- Specify your Freesurfer license

Connectome Mapper 3 BIDS App GUI

BIDS App Version
Tag: v3.0.0-RC3

BIDS dataset
Input directory: /home/localadmin/Desktop/hcp-retest-d2
Output directory: /home/localadmin/Desktop/hcp-retest-d2/derivatives

Participant labels to be processed

Available labels	Labels to be processed
sub-103818	sub-103818

Parallel processing Number of participants processed in parallel: 1

Advanced execution settings for each participant process

Multithreading Random number generators

Number of OpenMP threads: 1 Set seed of ANTS random number generator: ☒ Seed: 1234

Set number of threads used by ANTs: ☒ Set seed of MRtrix random number generator: ☒ Seed: 1234

Number of ITK threads used by ANTs registration: 1 Set seed of MRtrix random number generator: ☒ Seed: 1234

Configuration of processing pipelines

Anatomical pipeline
Configuration file: /home/localadmin/Desktop/hcp-retest-d2/code/ref_anatomical_config.ini

Diffusion pipeline
Run processing stages: ☒
Configuration file: /home/localadmin/Desktop/hcp-retest-d2/code/ref_diffusion_config.ini

fMRI pipeline
Run processing stages: ☐

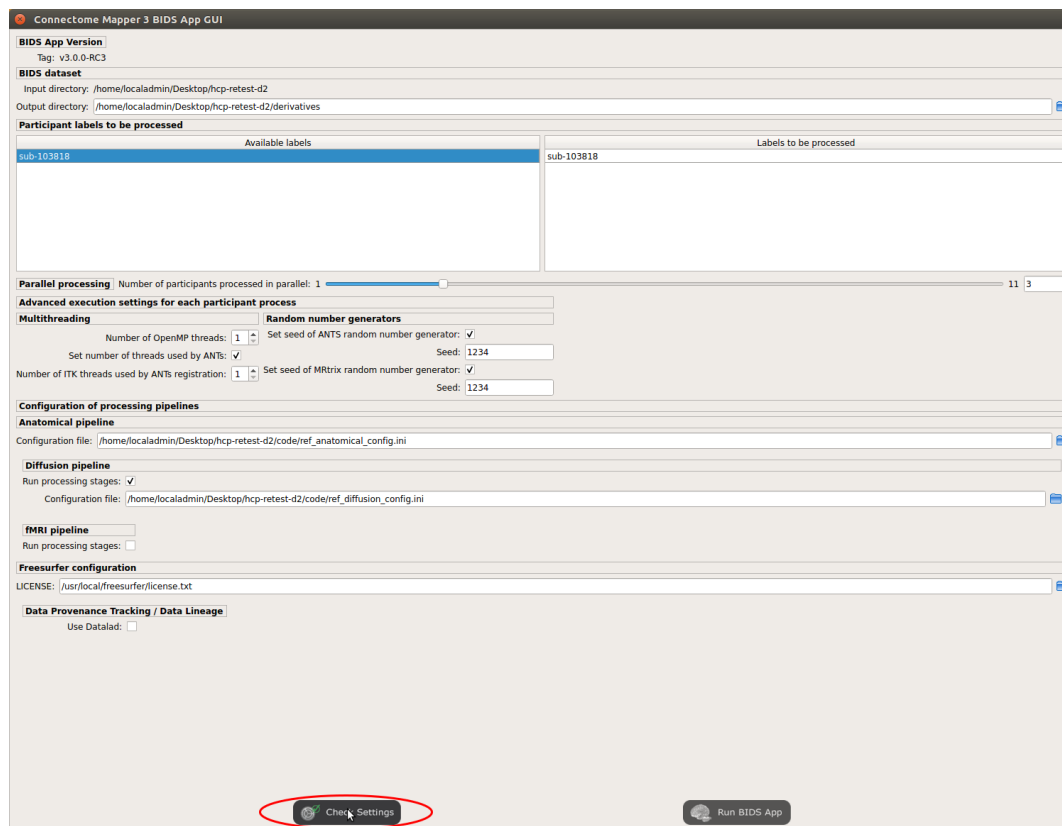
FreeSurfer configuration
LICENSE: /usr/local/freesurfer/license.txt

Data Provenance Tracking / Data Lineage Specifies Path to your FREESURFER license.txt
Use DataLad: ☐

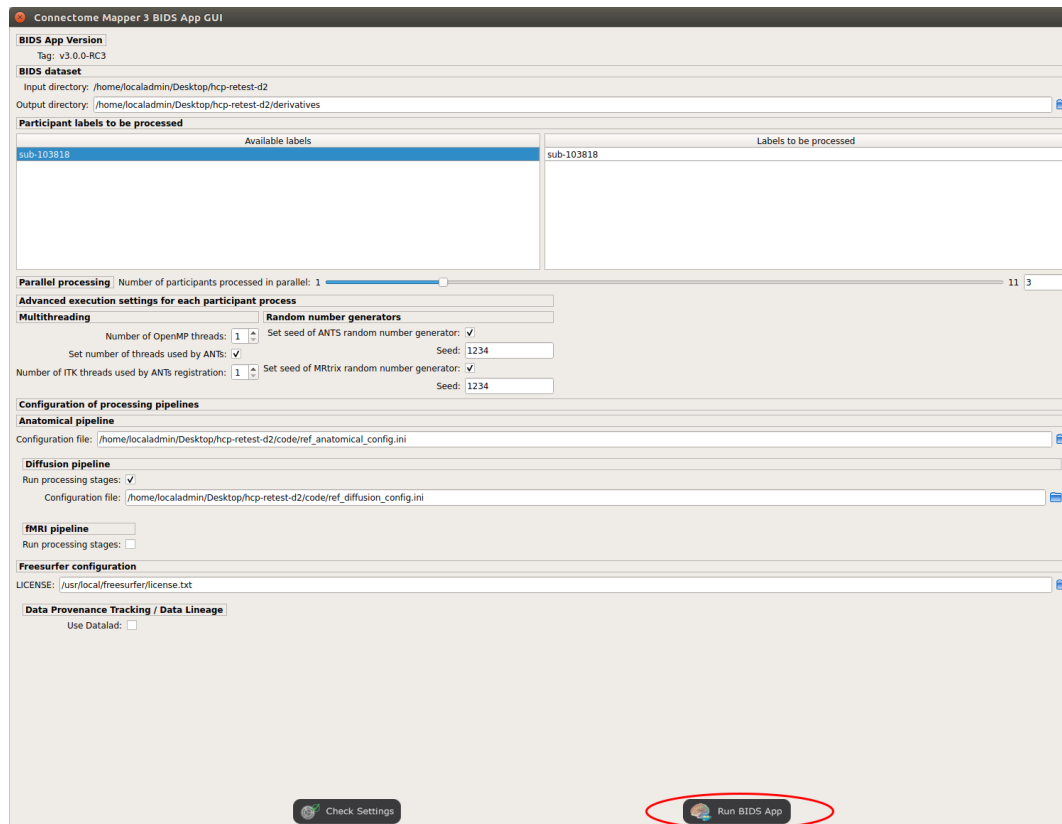
Buttons: Check Settings, Run BIDS App

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.



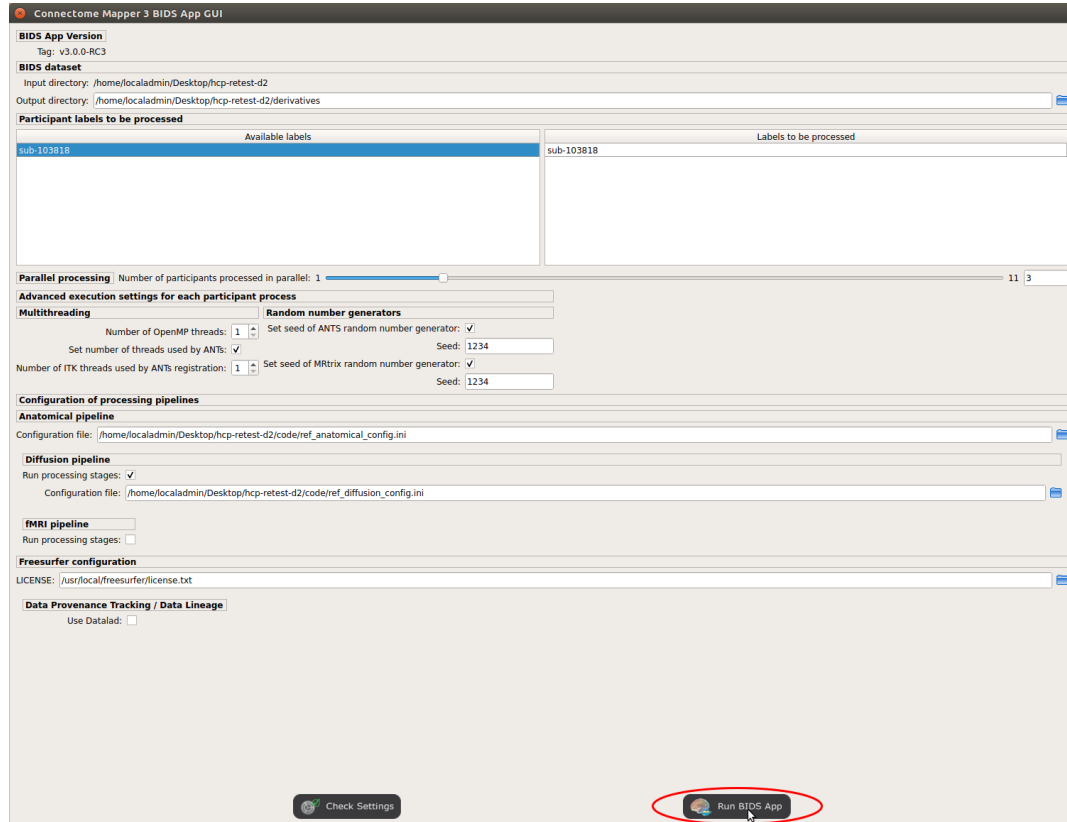
- If the setup is complete and valid, this will enable the Run BIDS App button.



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



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

```
Start BIDS App
> FreeSurfer license copy skipped as it already exists (BIDS App Manager)
> Datalad available: True
... BIDS App execution command: ['docker', 'run', '-it', '--rm', '-v', '/home/localadmin/Desktop/hcp-retest-d2:/bids_dir', '-v', '/home/localadmin/Desktop/hcp-retest-d2/derivatives:/output_dir', '-v', '/usr/local/freesurfer/license.txt:/bids_dir/code/license.txt', '-v', '/home/localadmin/Desktop/hcp-retest-d2/code/ref_anatomical_config.ini:/code/ref_anatomical_config.ini', '-v', '/home/localadmin/Desktop/hcp-retest-d2/code/ref_diffusion_config.ini:/code/ref_diffusion_config.ini', '-v', '/home/localadmin/Desktop/hcp-retest-d2/code/ref_fMRI_config.ini:/code/ref_fMRI_config.ini', '-u', '1000:1000', 'sebastientourbier/connectomemapper-bidsapp:v3.0.0-RC3', '/bids_dir', '/output_dir', 'participant', '--participant_label', '103818', '--anat_pipeline_config', '/code/ref_anatomical_config.ini', '--dwi_pipeline_config', '/code/ref_diffusion_config.ini', '--func_pipeline_config', '/code/ref_fMRI_config.ini', '--fs_license', '/bids_dir/code/license.txt', '--number_of_participants_processed_in_parallel', '1', '--number_of_threads', '10', '--ants_number_of_threads', '10']
```

(continues on next page)

(continued from previous page)

```

> BIDS dataset: /bids_dir
> Subjects to analyze : ['103818']
> Set $FS_LICENSE which points to FreeSurfer license location (BIDS App)
... $FS_LICENSE : /bids_dir/code/license.txt
* Number of subjects to be processed in parallel set to 1 (Total of cores_
↳available: 11)
* Number of parallel threads set to 10 (total of cores: 11)
* OMP_NUM_THREADS set to 10 (total of cores: 11)
* ITK_GLOBAL_DEFAULT_NUMBER_OF_THREADS set to 10
Report execution to Google Analytics.
Thanks to support us in the task of finding new funds for CMP3 development!
> Sessions to analyze : ['ses-01']
> Process subject sub-103818 session ses-01
WARNING: rewriting config file /output_dir/cmp/sub-103818/ses-01/sub-103818_
↳ses-01_anatomical_config.ini
... Anatomical config created : /output_dir/cmp/sub-103818/ses-01/sub-
↳103818_ses-01_anatomical_config.ini
WARNING: rewriting config file /output_dir/cmp/sub-103818/ses-01/sub-103818_
↳ses-01_diffusion_config.ini
... Diffusion config created : /output_dir/cmp/sub-103818/ses-01/sub-103818_
↳ses-01_diffusion_config.ini
WARNING: rewriting config file /output_dir/cmp/sub-103818/ses-01/sub-103818_
↳ses-01_fMRI_config.ini
... fMRI config created : /output_dir/cmp/sub-103818/ses-01/sub-103818_ses-
↳01_fMRI_config.ini
... Running pipelines :
    - Anatomical MRI (segmentation and parcellation)
    - Diffusion MRI (structural connectivity matrices)
    - fMRI (functional connectivity matrices)
... cmd : connectomemapper3 --bids_dir /bids_dir --output_dir /output_dir --
↳participant_label sub-103818 --session_label ses-01 --anat_pipeline_
↳config /output_dir/cmp/sub-103818/ses-01/sub-103818_ses-01_anatomical_
↳config.ini --dwi_pipeline_config /output_dir/cmp/sub-103818/ses-01/sub-
↳103818_ses-01_diffusion_config.ini --func_pipeline_config /output_dir/cmp/
↳sub-103818/ses-01/sub-103818_ses-01_fMRI_config.ini --number_of_threads 10

```

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

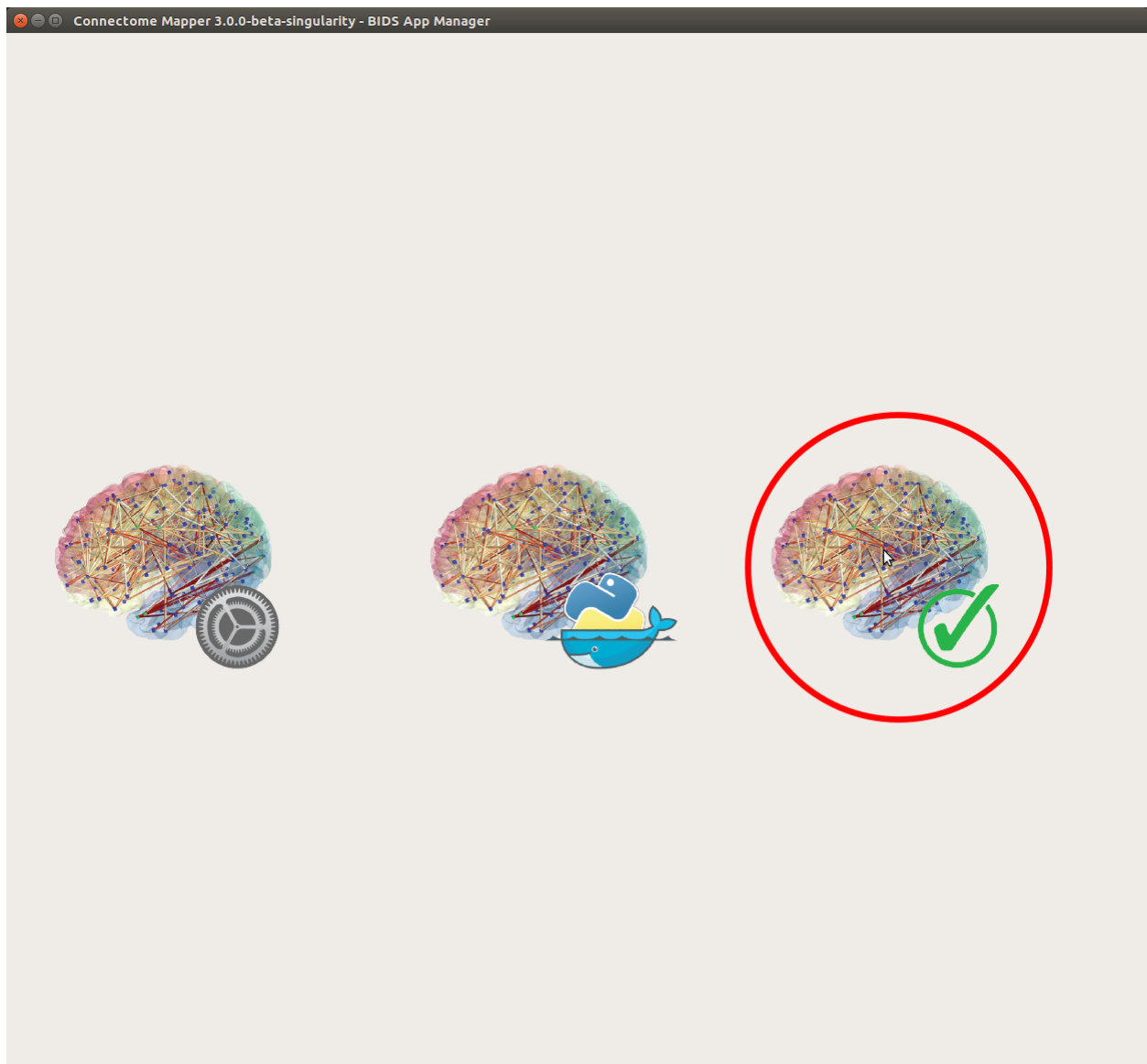
Check progress

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):

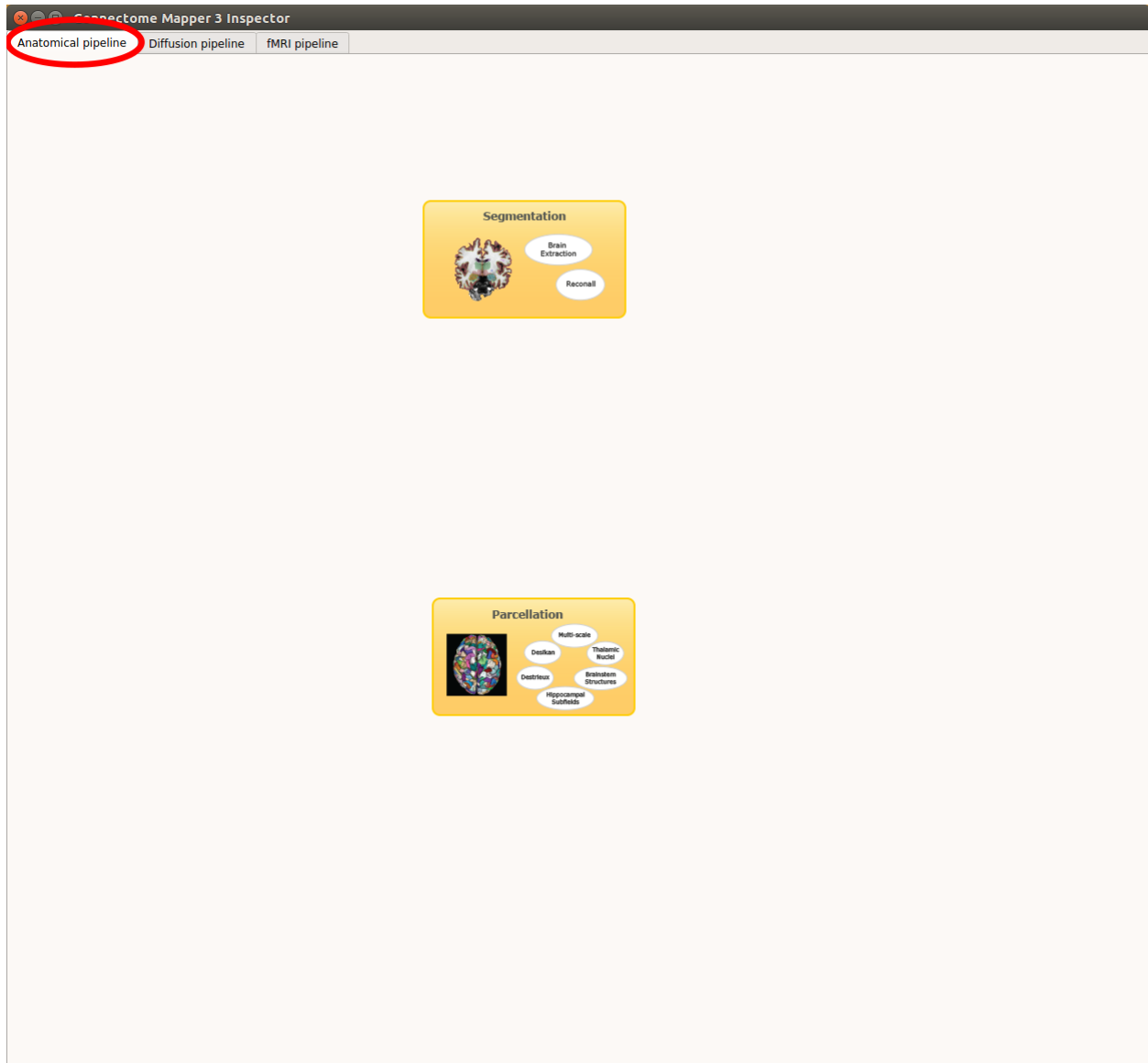
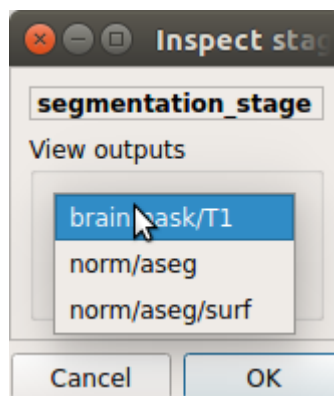


Fig. 8: Panel for configuration of anatomical pipeline stages

Segmentation

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



Segmentation results

Surfaces extracted using Freesurfer.

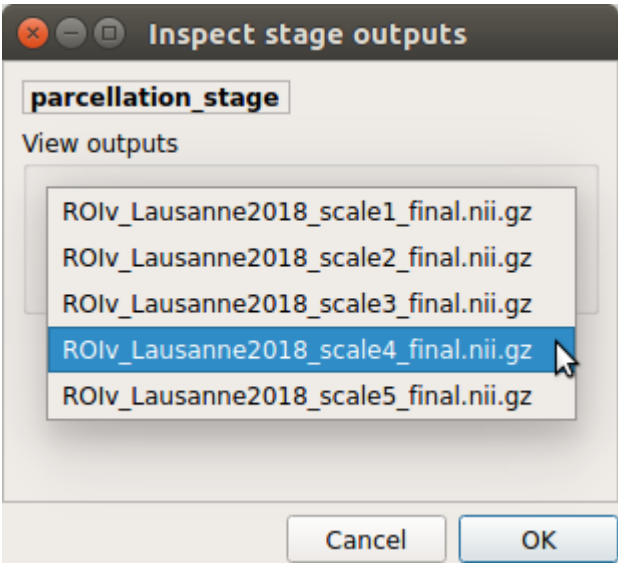


T1 segmented using Freesurfer.



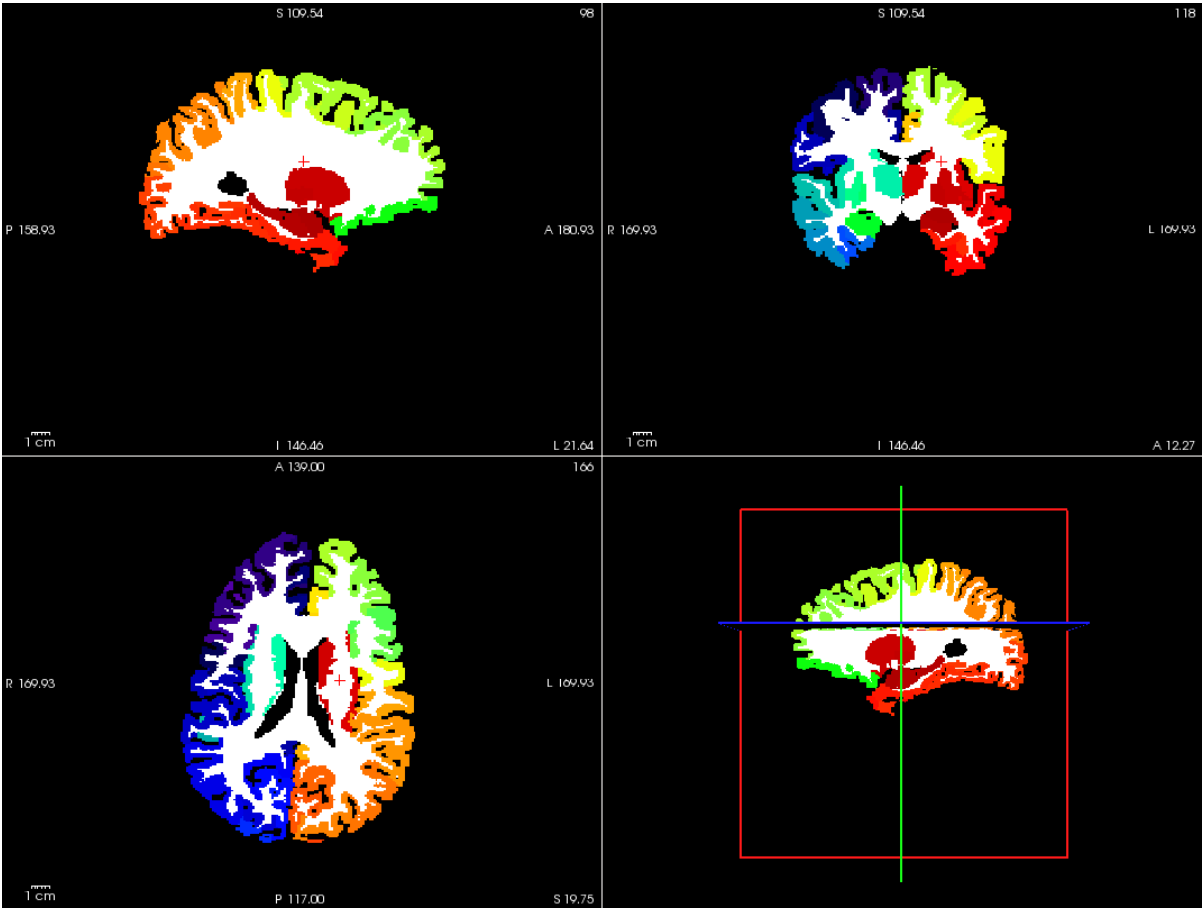
Parcellation

- 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):

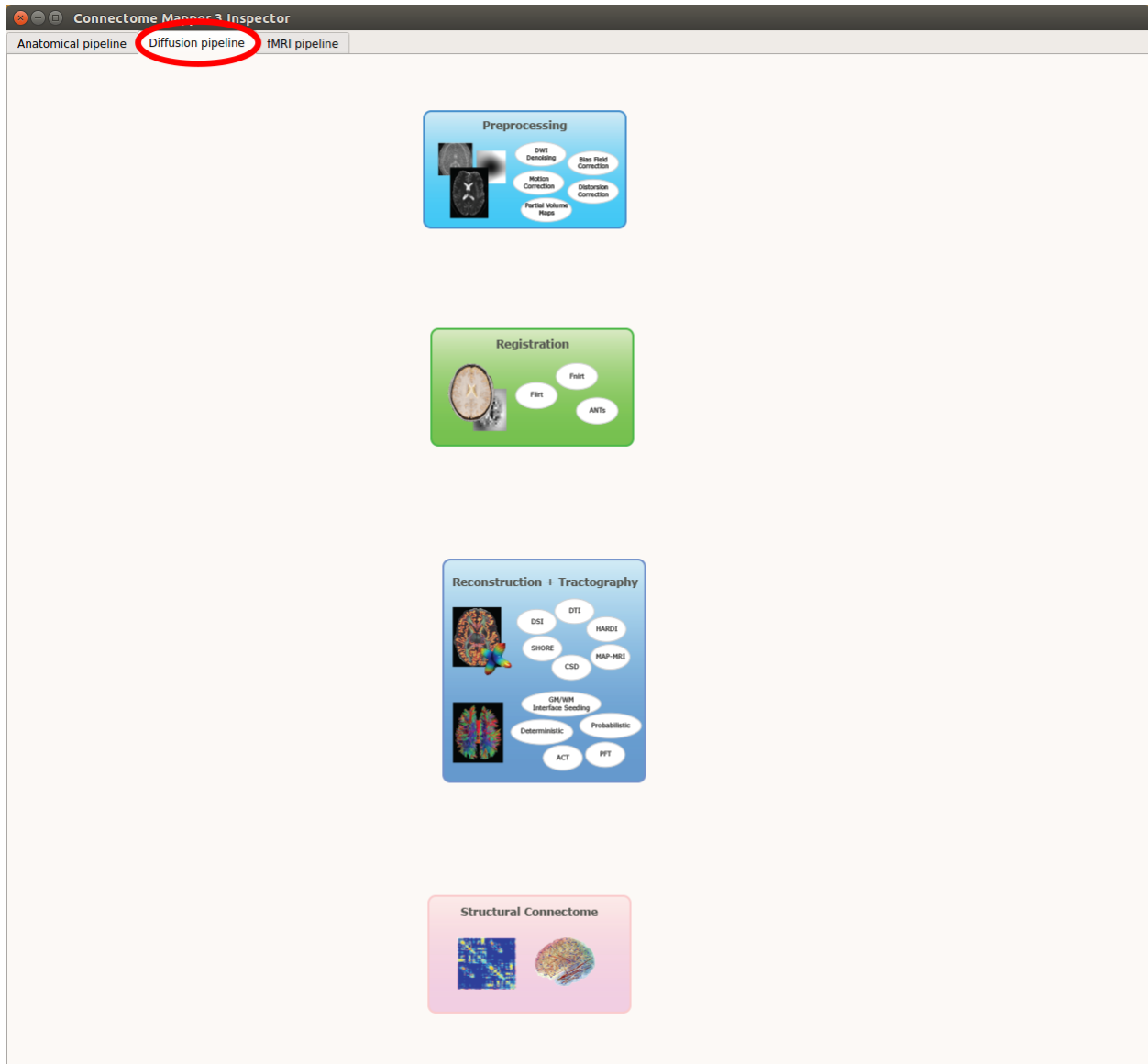
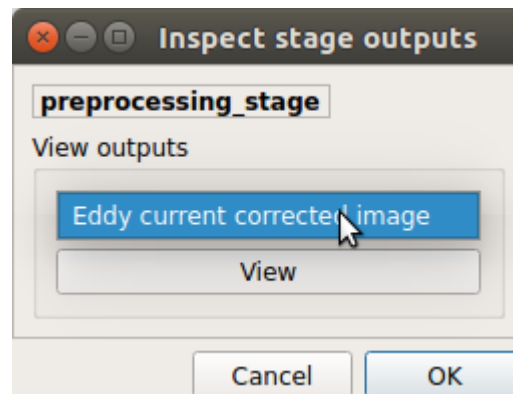


Fig. 9: Panel for configuration of diffusion pipeline stages

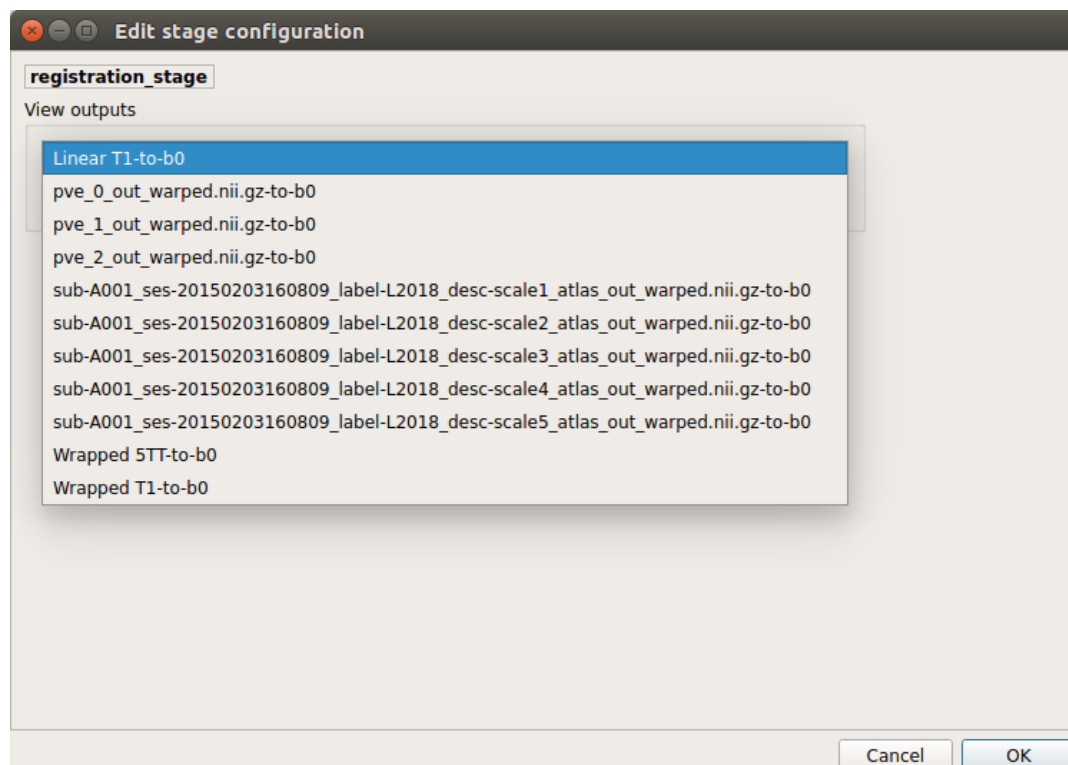
Preprocessing

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



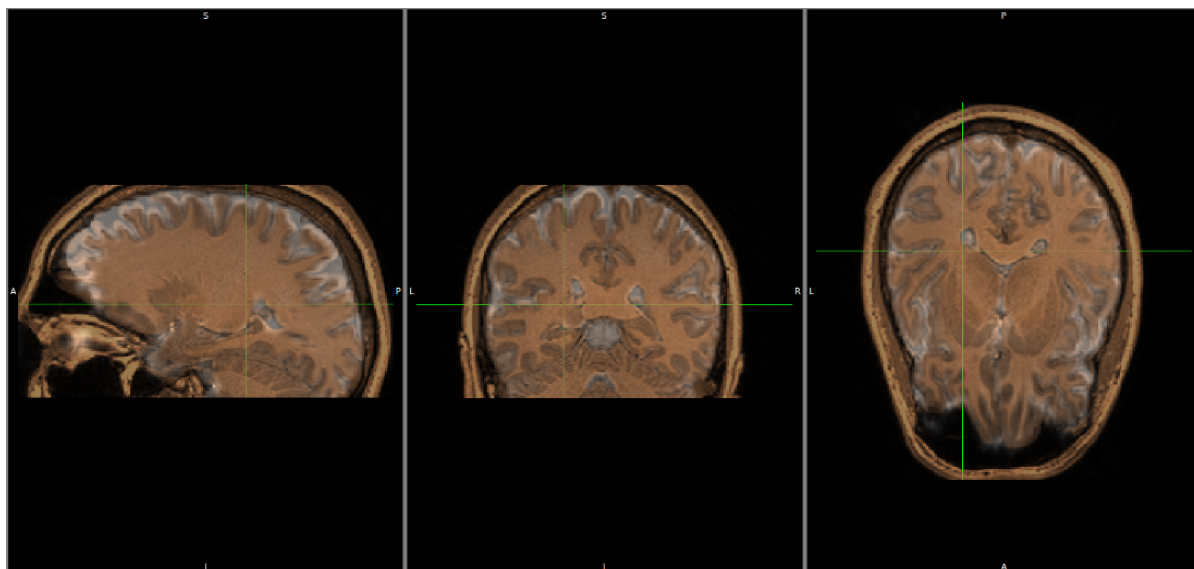
Registration

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



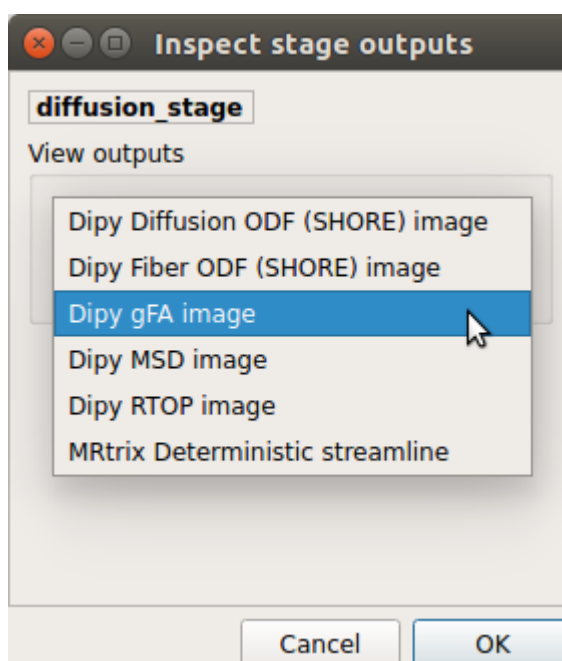
Registration results

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



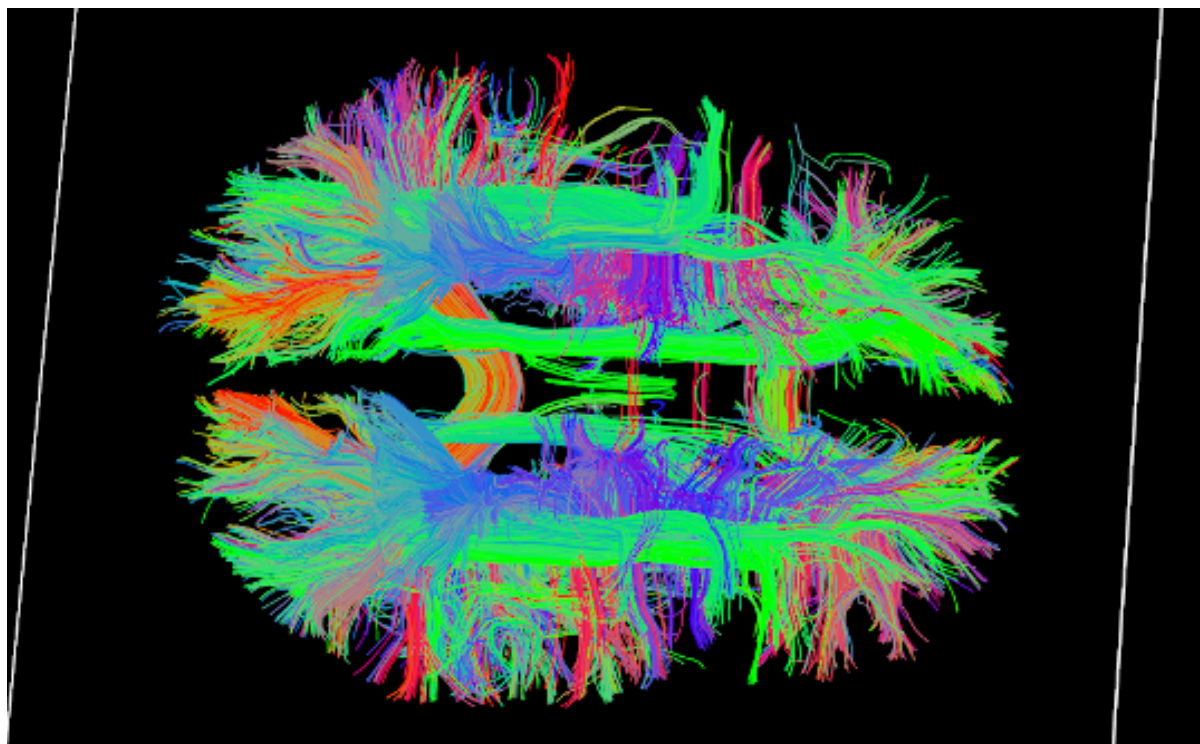
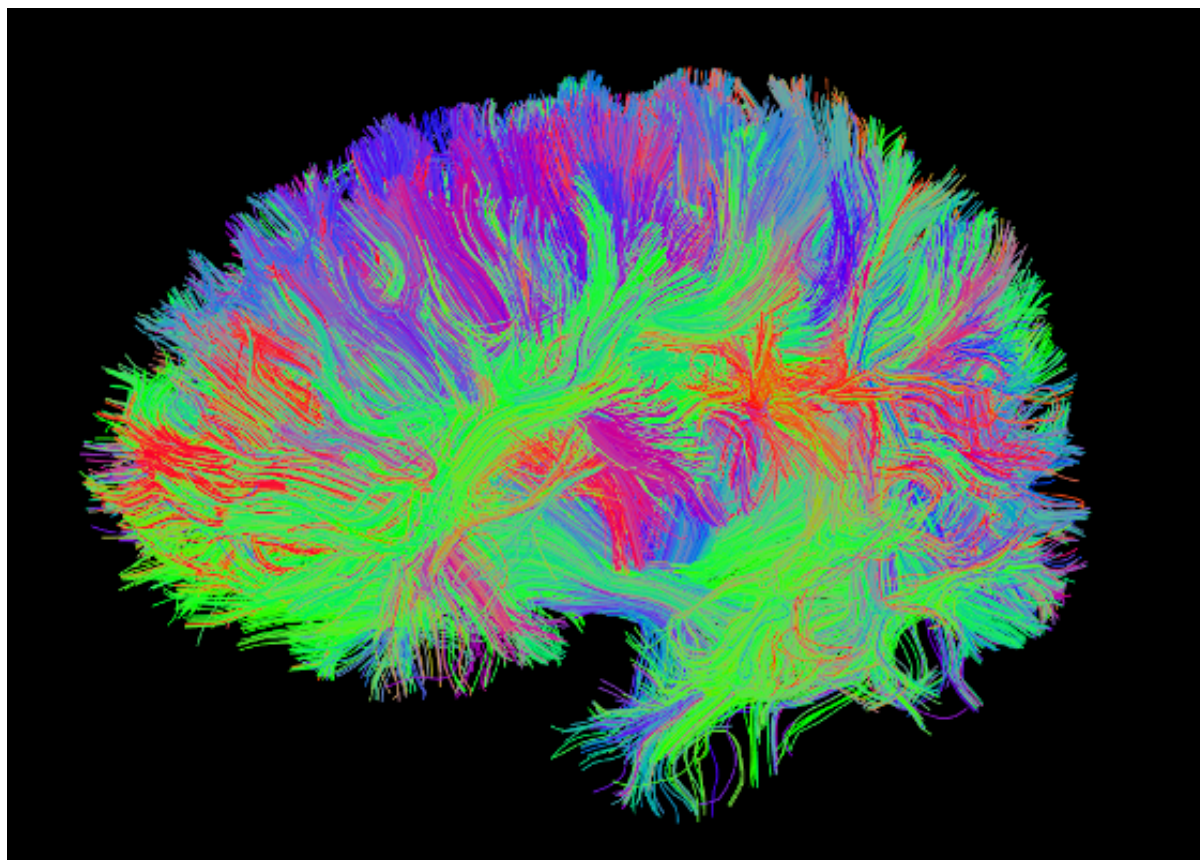
Diffusion reconstruction and tractography

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



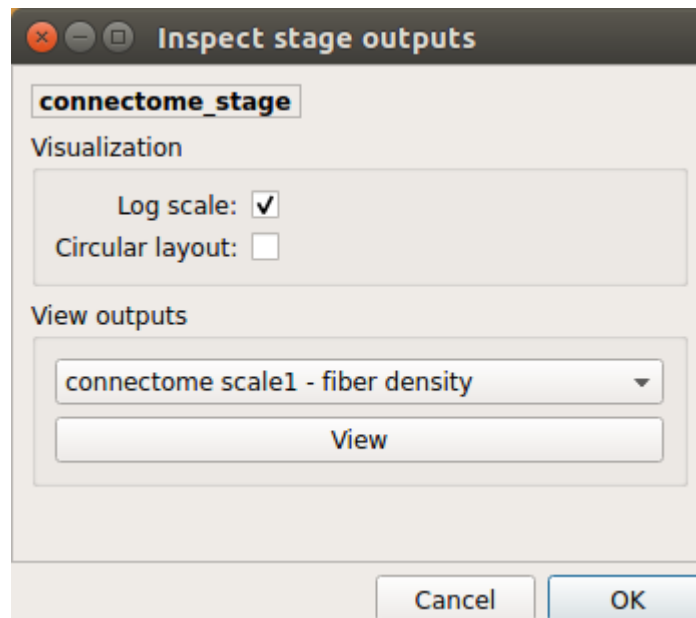
Tractography results

DSI Tractography results are displayed with TrackVis.



Connectome

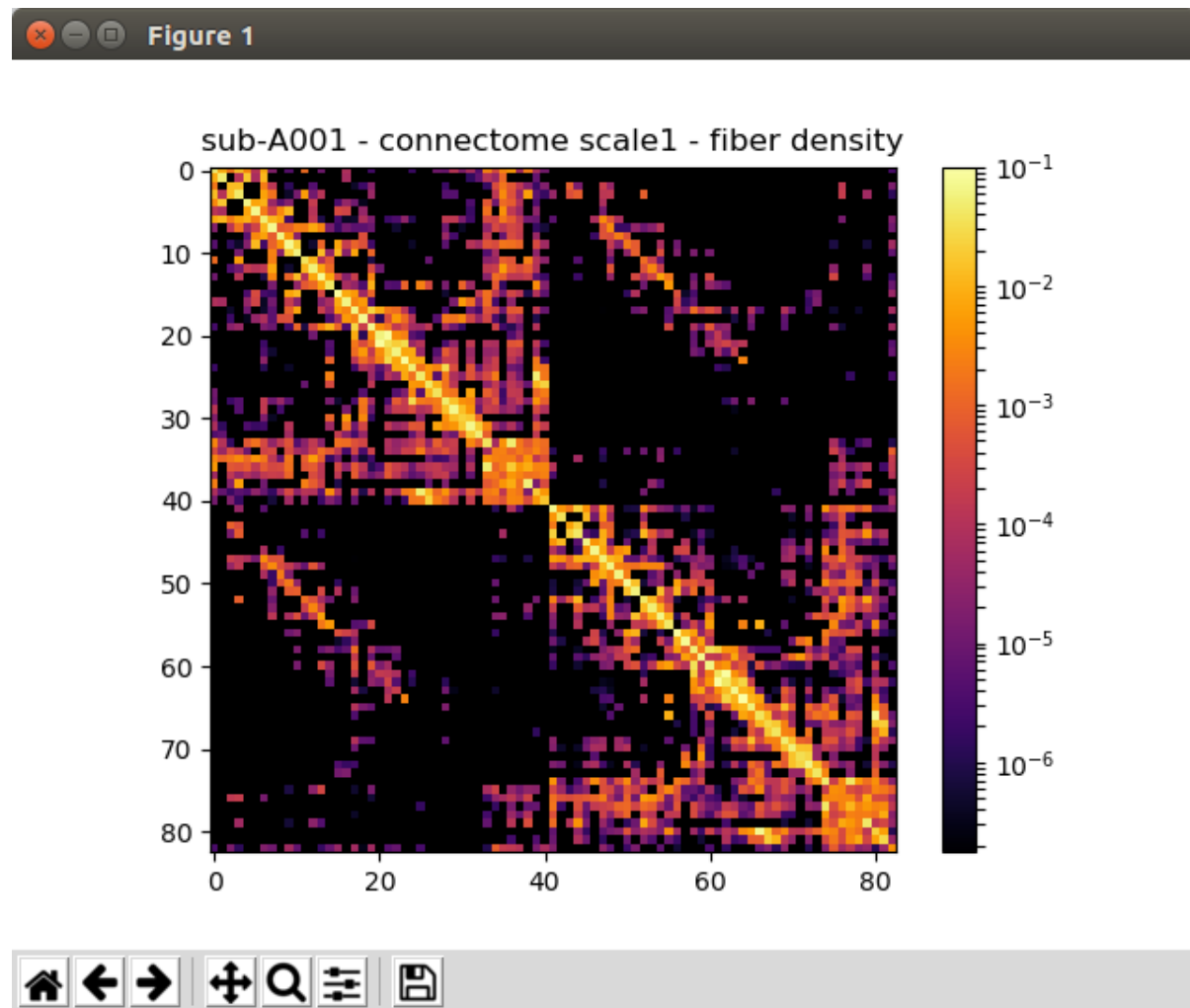
- Select the desired output from the list and click on view:



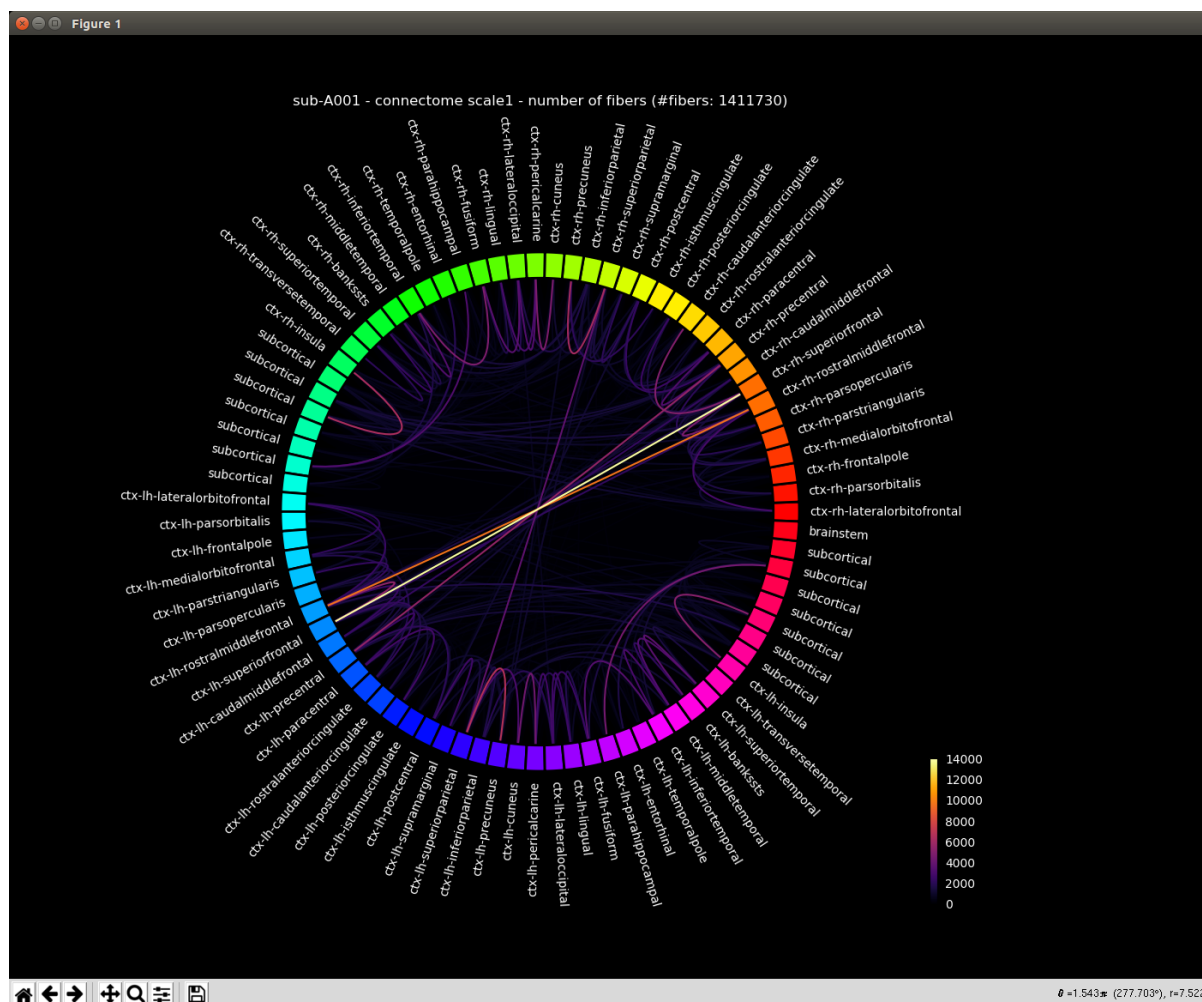
Generated connection matrix

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):

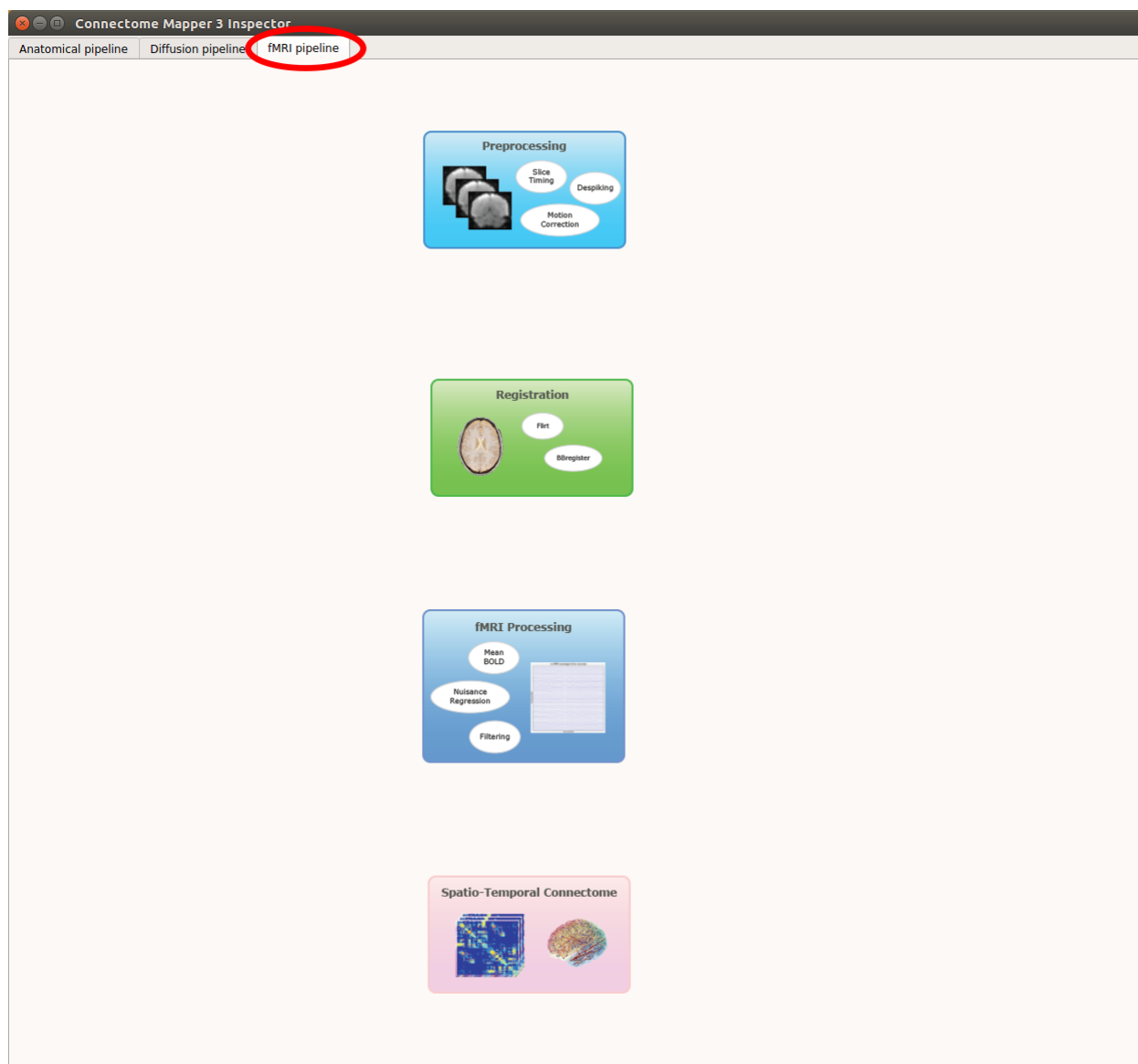
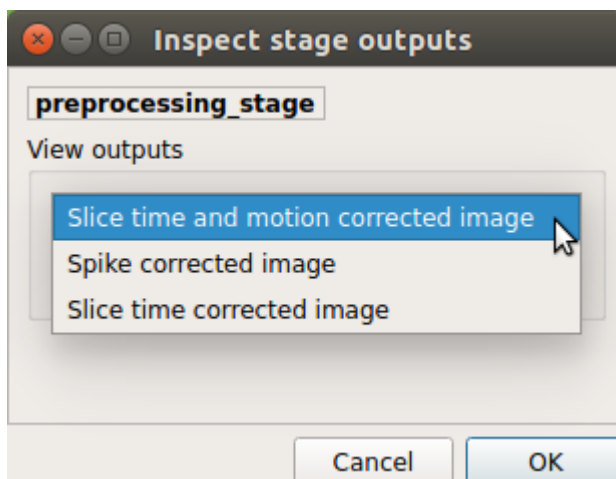


Fig. 10: Panel for configuration of fMRI pipeline stages

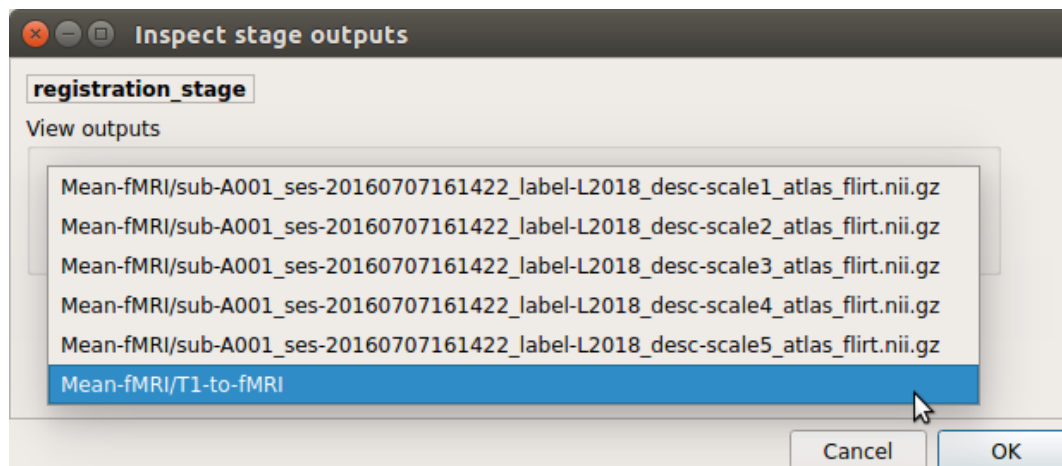
Preprocessing

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



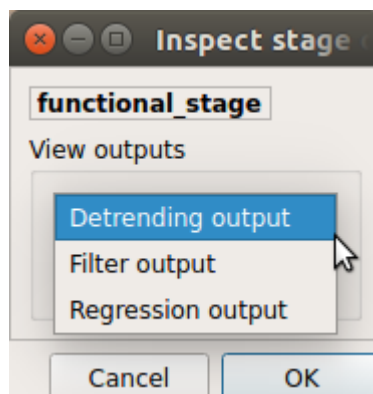
Registration

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

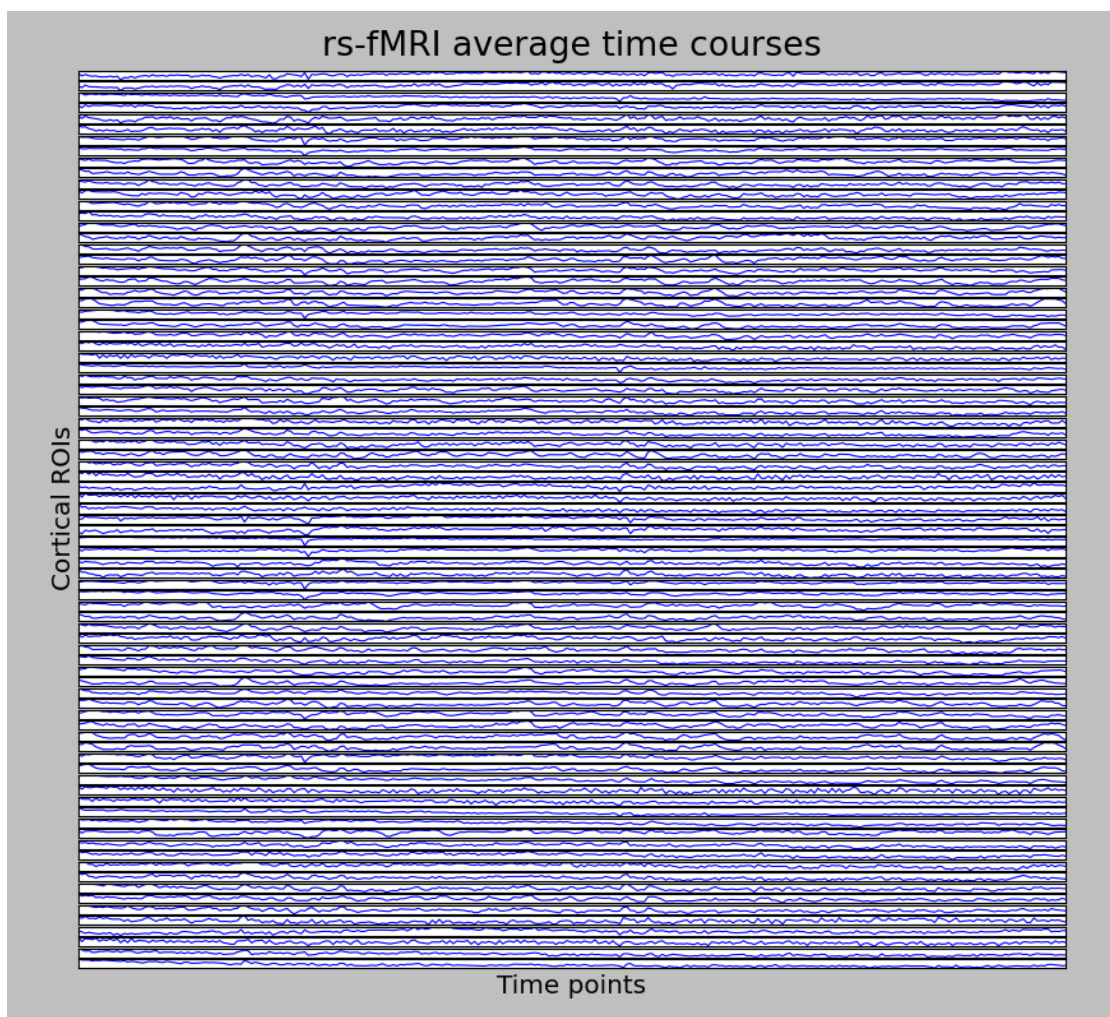


fMRI processing

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

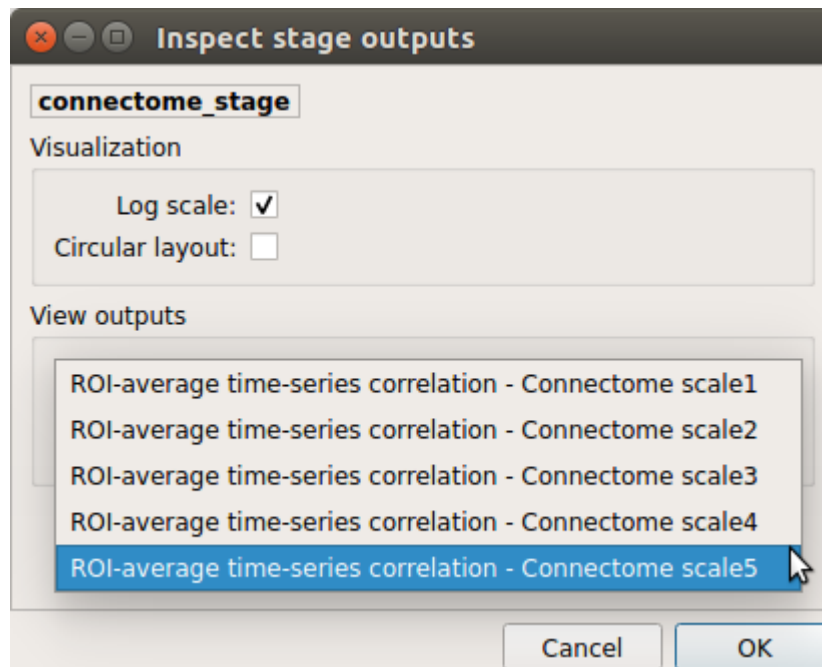


ROI averaged time-series



Connectome

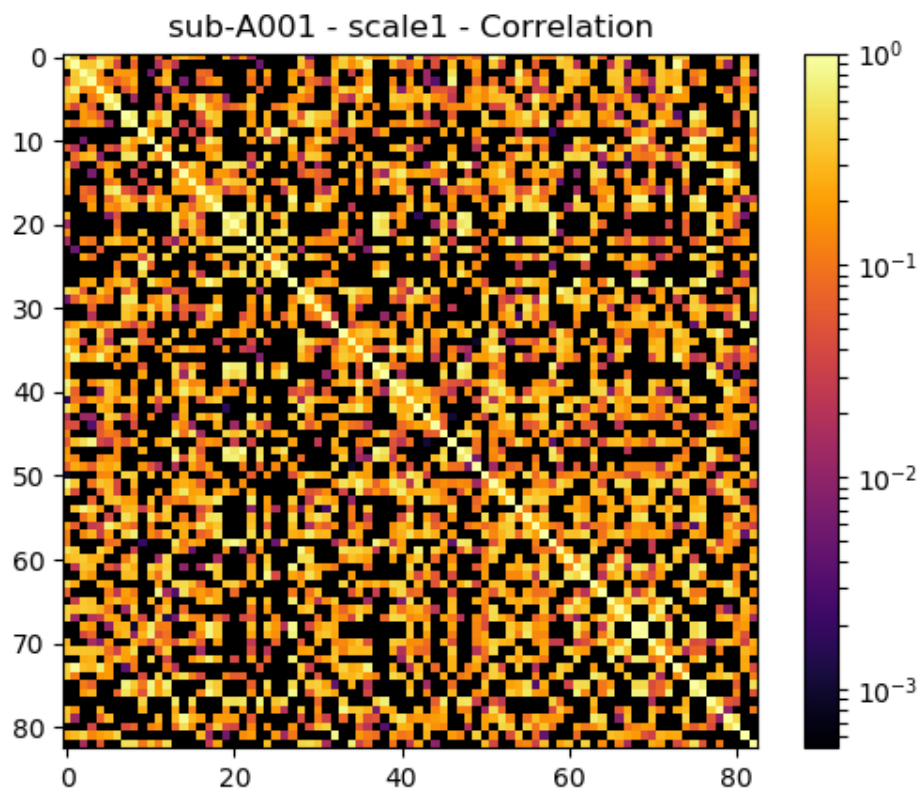
- Select the desired output from the list and click on view:



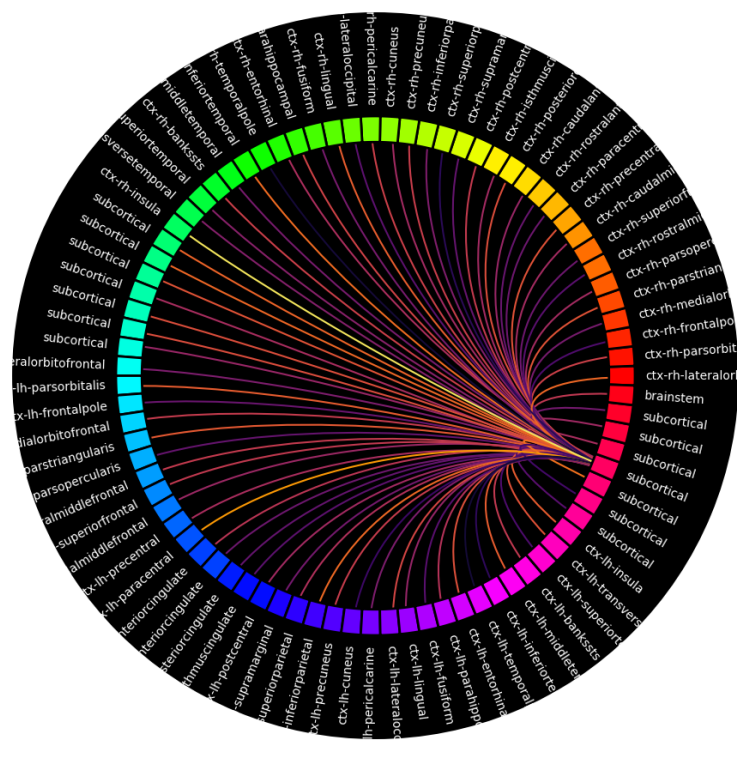
Generated connection matrix

Displayed using a:

1. matrix layout with pyplot



2. circular layout with pyplot and MNE



5.5 Outputs of Connectome Mapper 3

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

5.5.1 BIDS derivatives entities

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

See [Original BIDS Entities Appendix](#) for more description.

Note: Connectome Mapper 3 introduced a new BIDS entity `atlas-<atlas_label>` (where `<atlas_label>`: Desikan/ L2008/ L2018), that is used in combination with the `res-<atlas_scale>` (where `<atlas_scale>`: scale1 / scale2 / scale3 / scale4 / scale5) entity to distinguish data derived from different parcellation atlases and different scales.

5.5.2 Main Connectome Mapper Derivatives

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

Anatomical derivatives

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

- * anat/sub-<subject_label>_label-GM_dseg.nii.gz
- * anat/sub-<subject_label>_label-CSF_dseg.nii.gz
- The five different brain parcellations:
 - * anat/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_dseg.nii.gz
 - where:
 - <atlas_label>: Desikan / L2008 / L2018 is the parcellation scheme used
 - <scale_label>: scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale if applicable
 - with two tsv side-car files that follow the [BIDS derivatives](#), one describing the parcel label/index mapping (_dseg.tsv), one reporting volumetry of the different parcels (_stats.tsv), and two files used internally by CMP3, one describing the parcel labels in graphml format (dseg.graphml), one providing the color lookup table of the parcel labels in Freesurfer format which can be used directly in freeview (_FreeSurferColorLUT.txt):
 - * anat/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_dseg.tsv
 - * anat/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_stats.tsv
 - * anat/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_dseg.graphml
 - * anat/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_FreeSurferColorLUT.txt
- Anatomical derivatives in the DWI space produced by the diffusion pipeline are placed in each subject's anat/ subfolder, including:
 - The unmasked T1w image:
 - * anat/sub-<subject_label>_space-DWI_desc-head_T1w.nii.gz
 - The masked T1w image with its corresponding brain mask:
 - * anat/sub-<subject_label>_space-DWI_desc-brain_T1w.nii.gz
 - * anat/sub-<subject_label>_space-DWI_desc-brain_mask.nii.gz
 - The segmentation of WM tissue used for tractography seeding:
 - * anat/sub-<subject_label>_space-DWI_label-WM_dseg.nii.gz
 - The five different brain parcellation are saved as:
 - * anat/sub-<subject_label>_space-DWI_atlas-<atlas_label>[_res-<scale_label>]_dseg.nii.gz
 - where:
 - <atlas_label>: Desikan / L2008 / L2018 is the parcellation scheme used
 - <scale_label>: scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale if applicable
 - The 5TT image used for Anatomically Constrained 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-GWMI_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.gzwith 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.gzwith 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.gzwith derived Generalized Fractional Anisotropic (GFA), Mean Squared Displacement (MSD), Return-to-Origin Probability (RTOP) maps:
 - dwi/sub-<subject_label>]_model-SHORE_GFA.nii.gz
 - dwi/sub-<subject_label>]_model-SHORE_MSD.nii.gz
 - dwi/sub-<subject_label>]_model-SHORE_RTOP.nii.gz

- The tractogram:
 - `dwi/sub-<subject_label>_model-<model_label>_desc-<label>_tractogram.trk`
 where:
 - * `<model_label>` is the diffusion model used to drive tractography (DTI, CSD, SHORE)
 - * `<label>` is the type of tractography algorithm employed (DET for deterministic, PROB for probabilistic)
- The structural connectivity (SC) graphs:
 - `dwi/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_conndata-network_connectivity.<fmt>`
 where:
 - * `<atlas_label>`: Desikan / L2008 / L2018 is the parcellation scheme used
 - * `<scale_label>`: scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale if applicable
 - * `<fmt>`: mat / gpickle / tsv / graphml is the format used to store the graph

Functional derivatives

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

- The original BOLD image:
 - `func/sub-<subject_label>_task-rest_desc-cmp_bold.nii.gz`
- The mean BOLD image:
 - `func/sub-<subject_label>_meanBOLD.nii.gz`
- The fully preprocessed band-pass filtered used to compute ROI time-series:
 - `func/sub-<subject_label>_desc-bandpass_task-rest_bold.nii.gz`
- For scrubbing (if enabled):
 - The change of variance (DVARs):
 - * `func/sub-<subject_label>_desc-scrubbing_DVARs.npy`
 - The frame displacement (FD):
 - * `func/sub-<subject_label>_desc-scrubbing_FD.npy`
- Motion-related time-series:
 - `func/sub-<subject_label>_motion.tsv`
- The ROI time-series for each parcellation scale:
 - `func/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_timeseries.npy`
 - `func/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_timeseries.mat`
 where:
 - * `<atlas_label>`: Desikan / L2008 / L2018 is the parcellation scheme used
 - * `<scale_label>`: scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale if applicable

- The functional connectivity (FC) graphs:
 - `func/sub-<subject_label>_atlas-<atlas_label>[_res-<scale_label>]_conndata-network_connectivity<fmt>`
where:
 - * `<atlas_label>`: Desikan / L2008 / L2018 is the parcellation scheme used
 - * `<scale_label>`: scale1, scale2, scale3, scale4, scale5 corresponds to the parcellation scale if applicable
 - * `<fmt>`: mat / gpickle / tsv / graphml is the format used to store the graph

5.5.3 FreeSurfer Derivatives

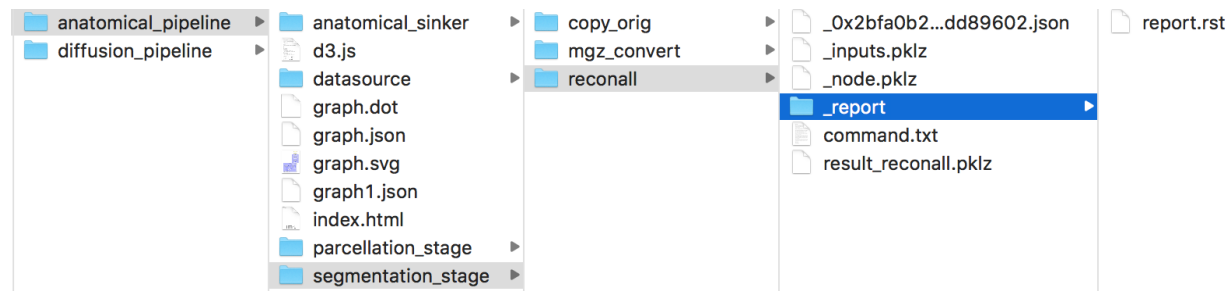
A FreeSurfer subjects directory is created in `<bids_dataset/derivatives>/freesurfer`.

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

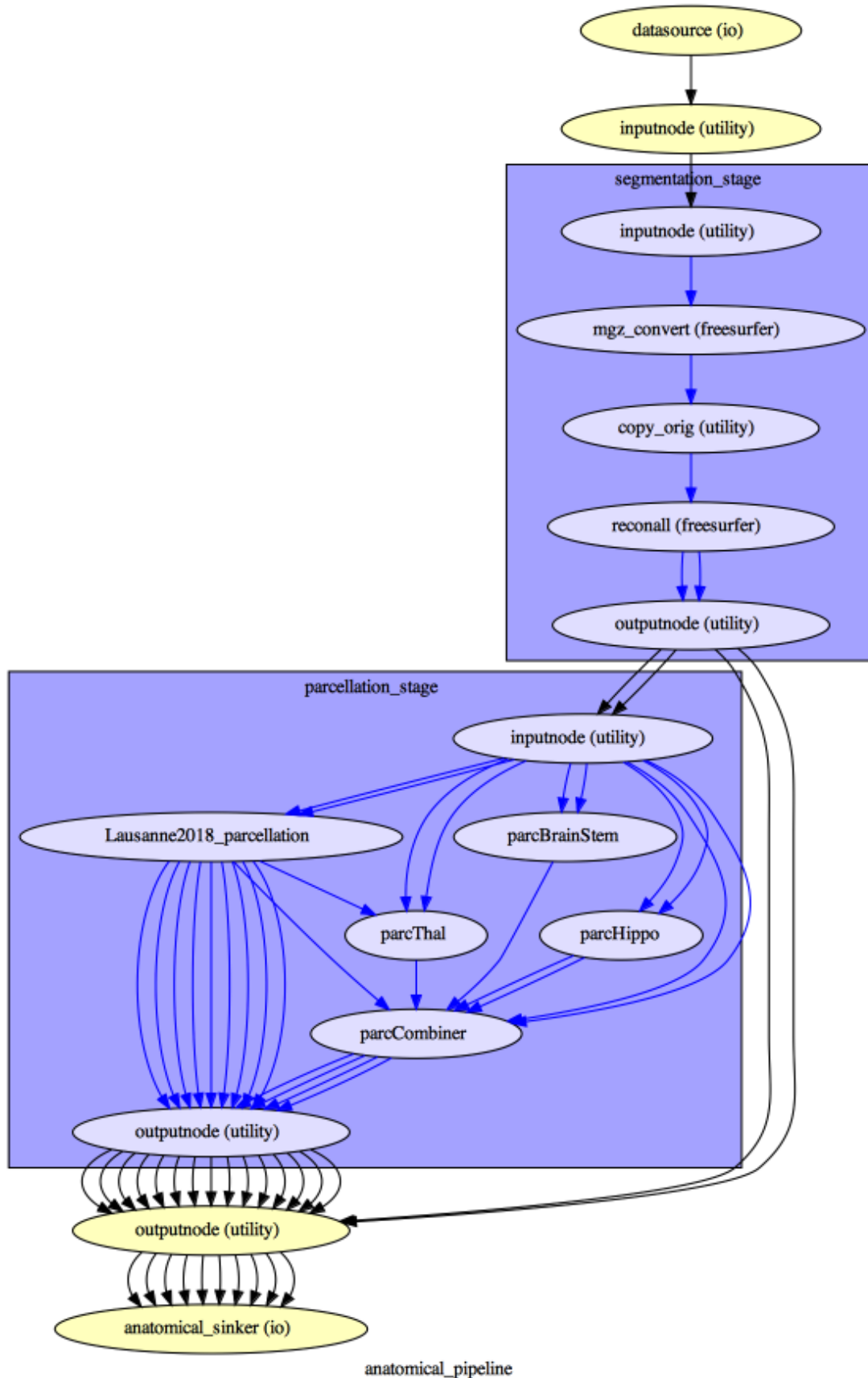
The fsaverage subject distributed with the running version of FreeSurfer is copied into this directory.

5.5.4 Nipype Workflow Derivatives

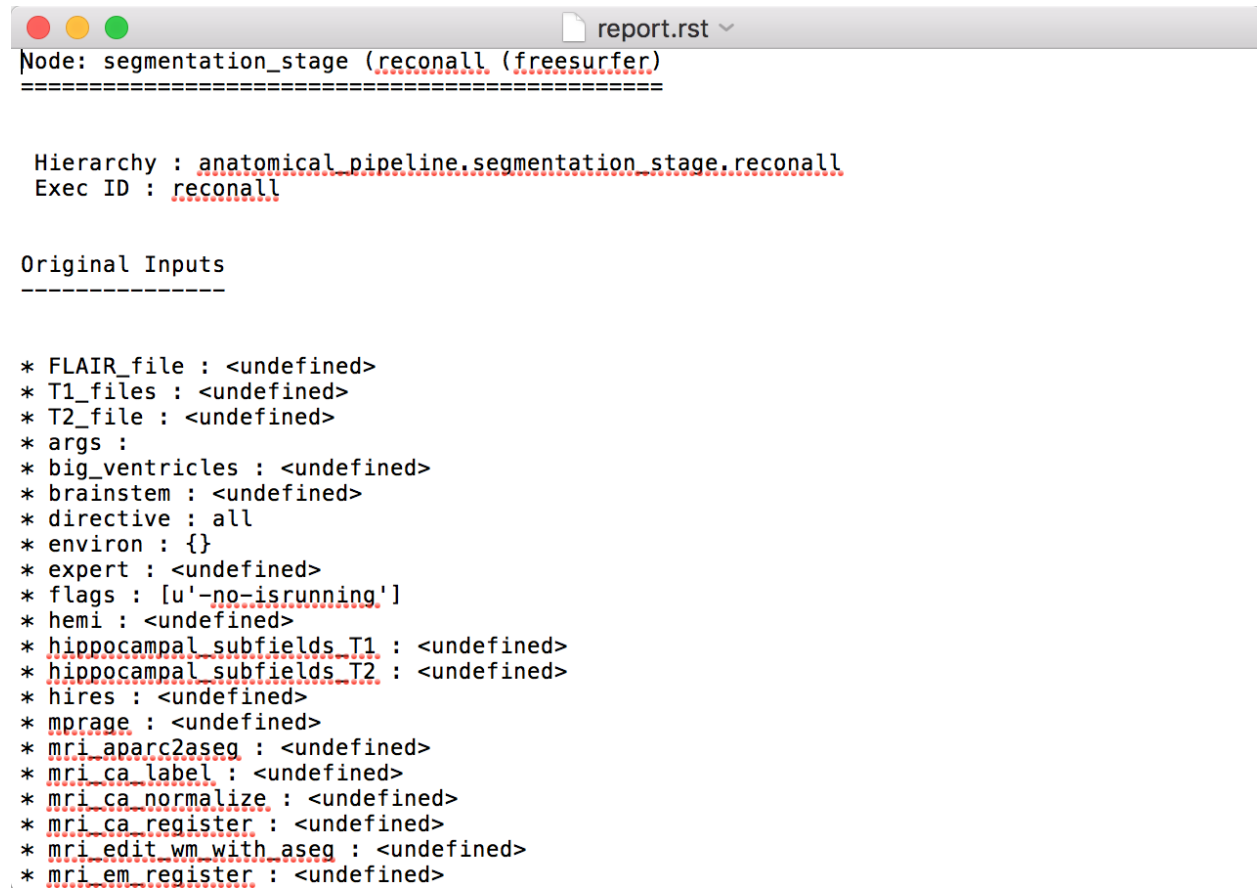
The execution of each Nipype workflow (pipeline) dedicated to the processing of one modality (i.e. anatomical/diffusion/fMRI) involves the creation of a number of intermediate outputs which are written to `<bids_dataset/derivatives>/nipype/sub-<subject_label>/<anatomical/diffusion/fMRI>_pipeline` respectively:



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 involved 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`



```
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.parser.get_docker_wrapper_parser()`

Return the argparse parser of the Docker BIDS App.

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

Return type `object`

`cmp.parser.get_singularity_wrapper_parser()`

Return the argparse parser of the Singularity 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', 'HARDI' or 'multishell'

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: 'Lausanne2018')

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-<version>'/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).

check_stages_execution()

Check stage execution.

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 inputs of the anatomical pipeline are available.

Parameters

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

Returns valid_inputs – True if inputs are available

Return type traits.Bool

check_output()

Check if outputs of an *AnatomicalPipeline* are available.

Returns

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

create_datagrabber_node(base_directory)

Create the appropriate Nipype DataGrabber node.

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

Returns *datasource* – Output Nipype Node with DataGrabber interface

Return type Output Nipype DataGrabber Node

create_datasinker_node(base_directory)

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

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

Returns *sinker* – Output Nipype Node with DataSink interface

Return type Output Nipype DataSink Node

create_pipeline_flow(cmp_deriv_subject_directory, nipype_deriv_subject_directory)

Create the pipeline workflow.

Parameters

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

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

Return type *nipype.pipeline.engine.Workflow*

define_custom_mapping(custom_last_stage)

Define the pipeline to be executed until a specific stages.

Not used yet by CMP3.

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

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

```
input_folders = ['anat']
```

```
now = '20211223_1027'
```

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

process()

Executes the anatomical pipeline workflow and returns True if successful.

```
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 'anatomical'
```

```
    subjects
```

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

```
        Type traits.List
```

```
    subject
```

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

```
        Type traits.Str
```

```
    subject_session
```

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

```
        Type traits.Str
```

cmp.pipelines.diffusion package

Submodules

cmp.pipelines.diffusion.diffusion module

Diffusion pipeline Class definition.

```
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_datagrabber_node(*base_directory, bids_atlas_label*)

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

Parameters

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

Returns `datasource` – Output Nipype Node with DataGrabber interface

Return type Output Nipype DataGrabber Node

create_datasinker_node(*base_directory, bids_atlas_label, recon_model, tracking_model*)

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

Parameters

- **base_directory** (*Directory*) – Main CMP output directory of a subject e.g. `/output_dir/cmp/sub-XX/(ses-YY)`
- **bids_atlas_label** (*string*) – Parcellation atlas label
- **recon_model** (*string*) – Diffusion signal model (DTI or CSD)
- **tracking_model** (*string*) – Tractography algorithm (DET or PROB)

Returns `sinker` – Output Nipype Node with DataSink interface

Return type Output Nipype DataSink Node

create_field_template_dict(*bids_atlas_label*)

Create the dictionary of input field template given to Nipype DataGrabber`

Parameters **bids_atlas_label** (*string*) – Parcellation atlas label

Returns `field_template` – Output dictionary of template input formats given to Nipype DataGrabber

Return type `dict`

create_pipeline_flow(*cmp_deriv_subject_directory, nipype_deriv_subject_directory*)

Create the pipeline workflow.

Parameters

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

Returns `diffusion_flow` – An instance of `nipype.pipeline.engine.Workflow`

Return type `nipype.pipeline.engine.Workflow`

define_custom_mapping(*custom_last_stage*)

Define the pipeline to be executed until a specific stages.

Not used yet by CMP3.

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

get_file(*layout, subject, suffix, extensions, session=None*)

Query files with PyBIDS and take the first file in the returned list or get a specific dmri file if BIDS acq-keyword is used in filename.

Parameters

- **layout** (*Instance(BIDSLayout)*) – Instance of pybids BIDSLayout
- **subject** (*str*) – BIDS subject/participant label i.e. XX in sub-XX
- **suffix** (*str*) – BIDS file suffix i.e. “T1w”, “dwi”, ...
- **extensions** (*str*) – File extension i.e. “.nii.gz”, “.json”, “.bval”, ...
- **session** (*str*) – BIDS session label i.e. YY in ses-YY if the dataset has multiple sessions

Returns **out_file** – The output filepath or None if no file was found

Return type *str*

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

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

```
now = '20211223_1027'
```

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

process()

Executes the diffusion 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** (*string*) – New value.

update_outputs_tracking(*new*)

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

Parameters **new** (*string*) – New value.

update_preprocessing_act(*new*)

Update self.stages["Preprocessing"].config.act_tracking when use_act is updated.

Parameters **new** (*string*) – New value.

update_preprocessing_gmwwi(*new*)

Update self.stages["Preprocessing"].config.gmwwi_seeding when seed_from_gmwwi is updated.

Parameters **new** (*string*) – New value.

update_tracking_tool(*new*)

Update self.stages["Preprocessing"].config.tracking_tool when tracking_processing_tool is updated.

Parameters **new** (*string*) – New value.

update_vizualization_layout(*new*)

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

Parameters **new** (*string*) – New value.

update_vizualization_logscale(*new*)

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

Parameters **new** (*bool*) – 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 red 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.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 (FunctionalMRIStage) that can perform nuisance regression and band-pass filtering
- the connectome stage that extracts the time-series of each parcellation ROI and computes the Pearson's correlation coefficient between ROI time-series to create the functional connectome.

See also:

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

check_config()

Check if the fMRI pipeline parameters is properly configured.

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

Return type string

check_input(layout, gui=True)

Check if input of the diffusion pipeline are available.

Parameters

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

Returns valid_inputs – True if inputs are available

Return type `traits.Bool`

create_datagrabber_node(base_directory, bids_atlas_label)

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

Parameters

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

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

Return type Output Nipype DataGrabber Node

create_datasinker_node(*base_directory, bids_atlas_label*)

Create the appropriate Nipype DataSink node depending on the parcellation_scheme

Parameters

- **base_directory** (*Directory*) – Main CMP output directory of a subject e.g. /output_dir/cmp/sub-XX/(ses-YY)
- **bids_atlas_label** (*string*) – Parcellation atlas label
- **recon_model** (*string*) – Diffusion signal model (DTI or CSD)
- **tracking_model** (*string*) – Tractography algorithm (DET or PROB)

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

Return type Output Nipype DataSink Node

create_pipeline_flow(*cmp_deriv_subject_directory, nipype_deriv_subject_directory*)

Create the pipeline workflow.

Parameters

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

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

Return type `nipype.pipeline.engine.Workflow`

define_custom_mapping(*custom_last_stage*)

Define the pipeline to be executed until a specific stages.

Not used yet by CMP3.

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

global_conf = <`cmp.pipelines.functional.fMRI.Global_Configuration` object>

input_folders = ['anat', 'func']

now = '20211223_1027'

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

process()

Executes the fMRI pipeline workflow and returns True if successful.

update_nuisance_requirements()

Update nuisance requirements.

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

update_registration()

Configure the list of registration tools.

update_scrubbing()

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

cmp.stages package

Subpackages

cmp.stages.connectome package

Submodules

cmp.stages.connectome.connectome module

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

class cmp.stages.connectome.connectome.ConnectomeConfig

Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a [ConnectomeStage](#) instance.

compute_curvature

Compute fiber curvature (Default: False)

Type `traits.Bool`

output_types

Output connectome format

Type `['gPickle', 'mat', 'graphml']`

connectivity_metrics

Set of connectome maps to compute

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

log_visualization

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

Type `traits.Bool`

circular_layout

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

Type `traits.Bool`

subject

BIDS subject ID (in the form sub-XX)

Type `traits.Str`

See also:

[`cmp.stages.connectome.connectome.ConnectomeStage`](#)

class `cmp.stages.connectome.connectome.ConnectomeStage(bids_dir, output_dir)`

Bases: [`cmp.stages.common.Stage`](#)

Class that represents the connectome building stage of a [`DiffusionPipeline`](#).

create_workflow()

Create the workflow of the diffusion [`ConnectomeStage`](#)

See also:

[`cmp.pipelines.diffusion.diffusion.DiffusionPipeline`](#), [`cmp.stages.connectome.connectome.ConnectomeConfig`](#)

create_workflow(flow, inputnode, outputnode)

Create the stage workflow.

Parameters

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

define_inspect_outputs()

Update the `inspect_outputs` class attribute.

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

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', 'multishell']

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_gmwmi

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, output_dir)`

Bases: [*cmp.stages.common.Stage*](#)

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

create_workflow()

Create the workflow of the *FunctionalMRIStage*

See also:

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

create_workflow(flow, inputnode, outputnode)

Create the stage 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: ‘Lausanne2018’)

Type *traits.Str*

parcellation_scheme_editor

Choice of parcellation schemes

Type *traits.List*([‘NativeFreesurfer’, ‘Lausanne2018’, ‘Custom’])

include_thalamic_nuclei_parcellation

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

Type traits.Bool

ants_precision_type

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

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

segment_hippocampal_subfields

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

Type traits.Bool

segment_brainstem

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

Type traits.Bool

atlas_info

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

Type traits.Dict

custom_parcellation

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

Type traits.Instance(*CustomParcellationBIDSFile*)

See also:

cmp.stages.parcellation.parcellation.ParcellationStage

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

Bases: *cmp.stages.common.Stage*

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

create_workflow()

Create the workflow of the *ParcellationStage*

See also:

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

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

Create the stage workflow.

Parameters

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

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

create_workflow_custom(*flow, outputnode*)

Create the stage workflow when custom inputs are specified.

Parameters

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

define_inspect_outputs()

Update the `inspect_outputs` class attribute.

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

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*, *output_dir*)

Bases: [`cmp.stages.common.Stage`](#)

Class that represents the pre-registration preprocessing stage of a [`fMRIPipeline`](#) instance.

create_workflow()

Create the workflow of the [`PreprocessingStage`](#)

See also:

[`cmp.pipelines.functional.fMRI.fMRIPipeline`](#),
[`fmri_preprocessing.PreprocessingConfig`](#)

[`cmp.stages.preprocessing.`](#)

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

Definition of config and stage classes for diffusion MRI preprocessing.

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

Bases: `traits.has_traits.HasTraits`

Class used to store configuration parameters of a [`PreprocessingStage`](#) instance.

total_readout

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

Type `traits.Float`

description

Description (Default: 'description')

Type `traits.Str`

denoising

Perform diffusion MRI denoising (Default: False)

Type traits.Bool

denoising_algo

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

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

dipy_noise_model

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

Type traits.Enum

bias_field_correction

Perform diffusion MRI bias field correction (Default: False)

Type traits.Bool

bias_field_algo

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

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

eddy_current_and_motion_correction

Perform eddy current and motion correction (Default: True)

Type traits.Bool

eddy_correction_algo

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

Type traits.Enum

eddy_correct_motion_correction

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

Type traits.Bool

partial_volume_estimation

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

Type traits.Bool

fast_use_priors

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

Type traits.Bool

resampling

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

Type traits.Tuple

interpolation

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

Type traits.Enum

tracking_tool

Tool used for tractography

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

act_tracking

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

Type Bool

gmwmi_seeding

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

Type Bool

See also:

[*cmp.stages.preprocessing.preprocessing.PreprocessingStage*](#)

class `cmp.stages.preprocessing.preprocessing.PreprocessingStage(bids_dir, output_dir)`

Bases: [*cmp.stages.common.Stage*](#)

Class that represents the pre-registration preprocessing stage of a [*DiffusionPipeline*](#) instance.

create_workflow()

Create the workflow of the [*PreprocessingStage*](#)

See also:

[*cmp.pipelines.diffusion.diffusion.DiffusionPipeline*](#), [*cmp.stages.preprocessing.preprocessing.PreprocessingConfig*](#)

create_workflow(*flow, inputnode, outputnode*)

Create the stage 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.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

tracking_tool

Tool used for tractography

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

act_tracking

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

Type traits.Bool

gmwmi_seeding

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

Type traits.Bool

See also:

[`cmp.stages.registration.registration.RegistrationStage`](#)

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

Bases: [`cmp.stages.common.Stage`](#)

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

fs_subjects_dir

Freesurfer subjects directory (needed by BBRegister)

Type `traits.Directory`

fs_subject_id

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

Type `traits.Str`

create_workflow()

Create the workflow of the [`RegistrationStage`](#)

See also:

[`cmp.pipelines.diffusion.diffusion.DiffusionPipeline`](#), [`cmp.pipelines.functional.fMRI.fMRIPipeline`](#), [`cmp.stages.registration.registration.RegistrationConfig`](#)

create_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.segmentation package

Submodules

cmp.stages.segmentation.segmentation module

Definition of config and stage classes for segmentation.

class cmp.stages.segmentation.segmentation.SegmentationConfig

Bases: traits.has_traits.HasTraits

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

seg_tool

Choice of segmentation tool that can be “Freesurfer”

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

make_isotropic

Resample to isotropic resolution (Default: False)

Type traits.Bool

isotropic_vox_size

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

Type traits.Float

isotropic_interpolation

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

Type traits.Enum

brain_mask_extraction_tool

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

Type traits.Enum

ants_templatefile

Anatomical template used by ANTS brain extraction

Type traits.File

ants_probmaskfile

Brain probability mask used by ANTS brain extraction

Type traits.File

ants_regmaskfile

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

Type traits.File

use_fsl_brain_mask

Use FSL BET for brain extraction (Default: False)

Type traits.Bool

use_existing_freesurfer_data

(Default: False)

Type traits.Bool

freesurfer_subjects_dir

Freesurfer subjects directory path usually /output_dir/freesurfer

Type traits.Str

freesurfer_subject_id

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

Type traits.Str

freesurfer_args

Extra Freesurfer recon-all arguments

Type traits.Str

custom_brainmask

Instance of [CustomBrainMaskBIDSFile](#) that describes the custom BIDS formatted brain mask

Type traits.Instance([CustomBrainMaskBIDSFile](#))

custom_wm_mask

Instance of [CustomWMMaskBIDSFile](#) that describes the custom BIDS formatted white-matter mask

Type traits.Instance([CustomWMMaskBIDSFile](#))

custom_gm_mask

Instance of [CustomGMMaskBIDSFile](#) that describes the custom BIDS formatted gray-matter mask

Type traits.Instance([CustomGMMaskBIDSFile](#))

custom_csf_mask

Instance of [CustomCSFMaskBIDSFile](#) that describes the custom BIDS formatted CSF mask

Type traits.Instance([CustomCSFMaskBIDSFile](#))

custom_aparcaseg

Instance of [CustomAparcAsegBIDSFile](#) that describes the custom BIDS formatted Freesurfer aparc-aseg file

Type traits.Instance([CustomAparcAsegBIDSFile](#))

number_of_threads

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

Type traits.Int

See also:

[cmp.stages.segmentation.segmentation.SegmentationStage](#), [cmtklib.bids.io.CustomBrainMaskBIDSFile](#), [cmtklib.bids.io.CustomWMMaskBIDSFile](#), [cmtklib.bids.io.CustomGMMaskBIDSFile](#), [cmtklib.bids.io.CustomCSFMaskBIDSFile](#)

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

Bases: [cmp.stages.common.Stage](#)

Class that represents the segmentation stage of a [AnatomicalPipeline](#).

create_workflow()

Create the workflow of the [SegmentationStage](#)

See also:

[cmp.pipelines.anatomical.anatomical.AnatomicalPipeline](#), [cmp.stages.segmentation.segmentation.SegmentationConfig](#)

create_workflow(*flow, inputnode, outputnode*)

Create the stage workflow.

Parameters

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

create_workflow_custom(*flow, inputnode, outputnode*)

Create the stage workflow when custom inputs are specified.

Parameters

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

define_inspect_outputs(*debug=False*)

Update the `inspect_outputs` class attribute.

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

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

has_run()

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

Returns

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

Submodules

cmp.stages.common module

Definition of common parent classes for stages.

class `cmp.stages.common.Stage`

Bases: `traits.has_traits.HasTraits`

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

It is extended by the various pipeline stage subclasses.

bids_subject_label

BIDS subject (participant) label

Type `traits.Str`

bids_session_label

BIDS session label

Type `traits.Str`

bids_dir

BIDS dataset root directory

Type traits.Str**output_dir**

Output directory

Type traits.Str**inspect_outputs**

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

Type traits.Dict**inspect_outputs_enum**

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

Type traits.Enum**enabled**

Stage enabled in the pipeline (Default: True)

Type traits.Bool**config**

Instance of stage configuration

Type Instance(HasTraits)**See also:**

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

enabled = True**inspect_outputs = ['Outputs not available']****is_running()**

Return the number of unfinished files in the stage.

Returns **nb_of_unfinished_files** – Number of unfinished files in the stage**Return type** int**GUI modules****cmp.bidsappmanager.gui module****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=False*)

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*)

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

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

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

Return type `bool`

check_output()

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

Returns

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

cmp.bidsappmanager.pipelines.diffusion package

Submodules

cmp.bidsappmanager.pipelines.diffusion.diffusion module

Diffusion pipeline UI Class definition.

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.fMRIPipelineUI`(*project_info*)

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

Class that extends the `fMRIPipeline` with graphical components.

preprocessing

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

Type `traits.ui.Button`

registration

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

Type `traits.ui.Button`

functionalMRI

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

Type `traits.ui.Button`

connectome

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

Type `traits.ui.Button`

view_mode

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

Type `['config_view', 'inspect_outputs_view']`

pipeline_group

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

Type `traitsUI panel`

traits_view

QtView that includes the `pipeline_group` panel

Type `QtView`

See also:

`cmp.pipelines.functional.fMRI.fMRIPipeline`

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

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

Parameters

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

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

Return type `bool`

cmp.bidsappmanager.stages package

Subpackages

cmp.bidsappmanager.stages.connectome package

Submodules

cmp.bidsappmanager.stages.connectome.connectome module

Definition of structural connectome config and stage UI classes.

class cmp.bidsappmanager.stages.connectome.connectome.ConnectomeConfigUI

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

Class that extends the ConnectomeConfig with graphical components.

output_types

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

Type list of string

connectivity_metrics

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

Type list of string

traits_view

TraitsUI view that displays the Attributes of this class

Type traits.ui.View

See also:

cmp.stages.connectome.connectome.ConnectomeConfig

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

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

Class that extends the ConnectomeStage with graphical components.

log_visualization

If True, display with a log transformation

Type traits.Bool

circular_layout

If True, display the connectivity matrix using a circular layout

Type traits.Bool

inspect_output_button

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

Type traits.ui.Button

inspect_outputs_view

TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view

TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

[*cmp.stages.connectome.connectome.ConnectomeStage*](#)

cmp.bidsappmanager.stages.connectome.fmri_connectome module

Definition of functional connectome config and stage UI classes.

class cmp.bidsappmanager.stages.connectome.fmri_connectome.ConnectomeConfigUI

Bases: [*cmp.stages.connectome.fmri_connectome.ConnectomeConfig*](#)

Class that extends the ConnectomeConfig with graphical components.

output_types

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

Type list of string

traits_view

TraitsUI view that displays the Attributes of this class

Type traits.ui.View

See also:

[*cmp.stages.connectome.fmri_connectome.ConnectomeConfig*](#)

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

Bases: [*cmp.stages.connectome.fmri_connectome.ConnectomeStage*](#)

Class that extends the ConnectomeStage with graphical components.

log_visualization

If True, display with a log transformation

Type traits.Bool

circular_layout

If True, display the connectivity matrix using a circular layout

Type traits.Bool

inspect_output_button

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

Type traits.ui.Button

inspect_outputs_view

TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view

TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

cmp.stages.connectome.fmri_connectome.ConnectomeStage

cmp.bidsappmanager.stages.diffusion package

Submodules

cmp.bidsappmanager.stages.diffusion.diffusion module

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, output_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*,
output_dir)

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

Class that extends the `FunctionalMRIStage` with graphical components.

inspect_output_button

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

Type `traits.ui.Button`

inspect_outputs_view

TraitsUI view that displays the quality inspection window of this stage

Type `traits.ui.View`

config_view

TraitsUI view that displays the configuration window of this stage

Type `traits.ui.View`

See also:

`cmp.stages.functional.functionalMRI.FunctionalMRIStage`

cmp.bidsappmanager.stages.parcellation package

Submodules

cmp.bidsappmanager.stages.parcellation.parcellation module

Definition of parcellation config and stage UI classes.

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

Bases: [cmp.stages.parcellation.parcellation.ParcellationConfig](#)

Class that extends the ParcellationConfig with graphical components.

custom_parcellation_view

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

Type traits.ui.View

traits_view

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

Type traits.ui.View

See also:

[cmp.stages.parcellation.parcellation.ParcellationConfig](#)

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

Bases: [cmp.stages.parcellation.parcellation.ParcellationStage](#)

Class that extends the ParcellationStage with graphical components.

inspect_output_button

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

Type traits.ui.Button

inspect_outputs_view

TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view

TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

[cmp.stages.parcellation.parcellation.ParcellationStage](#)

cmp.bidsappmanager.stages.preprocessing package

Submodules

cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing module

Definition of fMRI preprocessing config and stage UI classes.

class cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing.PreprocessingConfigUI
Bases: [cmp.stages.preprocessing.fmri_preprocessing.PreprocessingConfig](#)

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

traits_view

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

Type traits.ui.View

See also:

[cmp.stages.preprocessing.fmri_preprocessing.PreprocessingConfig](#)

class cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing.PreprocessingStageUI(*bids_dir*,
out-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-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="",
output_dir="")

Bases: [*cmp.stages.registration.registration.RegistrationStage*](#)

Class that extends the `RegistrationStage` with graphical components.

inspect_output_button

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

Type traits.ui.Button

inspect_outputs_view

TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view

TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

[*cmp.stages.registration.registration.RegistrationStage*](#)

cmp.bidsappmanager.stages.segmentation package**Submodules****cmp.bidsappmanager.stages.segmentation.segmentation module**

Definition of segmentation config and stage UI classes.

class cmp.bidsappmanager.stages.segmentation.segmentation.SegmentationConfigUI

Bases: [*cmp.stages.segmentation.segmentation.SegmentationConfig*](#)

Class that extends the SegmentationConfig with graphical components.

custom_brainmask_group

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

Type traits.ui.VGroup

custom_gm_mask_group

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

Type traits.ui.VGroup

custom_wm_mask_group

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

Type traits.ui.VGroup

custom_csf_mask_group

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

Type traits.ui.VGroup

custom_aparcaseg_group

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

Type traits.ui.VGroup

traits_view

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

Type traits.ui.View

See also:

[*cmp.stages.segmentation.segmentation.SegmentationConfig*](#)

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

Bases: [*cmp.stages.segmentation.segmentation.SegmentationStage*](#)

Class that extends the SegmentationStage with graphical components.

inspect_output_button

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

Type traits.ui.Button

inspect_outputs_view

TraitsUI view that displays the quality inspection window of this stage

Type traits.ui.View

config_view

TraitsUI view that displays the configuration window of this stage

Type traits.ui.View

See also:

[*cmp.stages.segmentation.segmentation.SegmentationStage*](#)

5.6.2 cmtklib package

Subpackages

cmtklib.bids package

Submodules

cmtklib.bids.io module

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

```
class cmtklib.bids.io.CustomAparcAsegBIDSFile
```

Bases: [*cmtklib.bids.io.CustomBIDSFile*](#)

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

```
class cmtklib.bids.io.CustomBIDSFile(p_toolbox_derivatives_dir="", p_suffix="", p_extension="",  
                                     p_acquisition="", p_atlas="", p_res="", p_label="", p_desc="")
```

Bases: traits.has_traits.HasTraits

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

toolbox_derivatives_dir

Toolbox folder name in the derivatives/ of the BIDS dataset

Type Str

suffix

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

Type Str

acquisition

Label used in _acq-<label>_

Type Str

res

Label used in _res-<label>_

Type Str

extension

File extension

Type Str

atlas

Label used in _atlas-<label>_

Type Str

label

Label used in _label-<label>_

Type Str

desc

Label used in _desc-<label>_

Type Str

get_filename_path(*base_dir*, *subject*, *session=None*, *debug=True*)

Return the number of regions by reading its associated TSV side car file describing the nodes.

Parameters

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

get_query_dict()

Return the dictionary to be passed to BIDSDataGrabber to query a list of files.

get_toolbox_derivatives_dir()

Return the value of custom_derivatives_dir attribute.

class cmcklib.bids.io.CustomBrainMaskBIDSFile

Bases: *cmcklib.bids.io.CustomBIDSFile*

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

class cmcklib.bids.io.CustomCSFMaskBIDSFile

Bases: *cmcklib.bids.io.CustomBIDSFile*

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

class cmcklib.bids.io.CustomGMMaskBIDSFile

Bases: *cmcklib.bids.io.CustomBIDSFile*

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

class `cmtklib.bids.io.CustomParcellationBIDSFile`

Bases: `cmtklib.bids.io.CustomBIDSFile`

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

get_nb_of_regions(*bids_dir*, *subject*, *session=None*, *debug=True*)

Return the number of regions by reading its associated TSV side car file describing the nodes.

Parameters

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

class `cmtklib.bids.io.CustomWMMaskBIDSFile`

Bases: `cmtklib.bids.io.CustomBIDSFile`

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

cmtklib.bids.utils module

This module provides CMTK Utility functions to handle BIDS datasets.

CreateBIDSStandardParcellationLabelIndexMappingFile

[Link to code](#)

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

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

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

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

verbose [a boolean] Verbose mode.

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

CreateCMPParcellationNodeDescriptionFilesFromBIDSFile

[Link to code](#)

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

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

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

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

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

CreateMultipleCMPParcellationNodeDescriptionFilesFromBIDSFile

[Link to code](#)

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

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

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

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

`cmtklib.bids.utils.get_native_space_files(filepathlist)`

Return a list of files without `_space-<label>_` in the filename.

`cmtklib.bids.utils.get_native_space_no_desc_files(filepathlist)`

Return a list of files without `_space-<label>_` and `_desc-<label>_` in the filename.

`cmtklib.bids.utils.get_native_space_tsv_sidecar_files(filepathlist)`

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

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

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

Parameters

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

cmtklib.interfaces package

Submodules

cmtklib.interfaces.afni module

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

Bandpass

[Link to code](#)

Bases: `nipy.interfaces.afni.base.AFNICCommand`

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] Despise 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 nipype.interfaces import afni
>>> despike = afni.Despike()
>>> despike.inputs.in_file = 'functional.nii'
>>> res = despike.run()
```

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

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

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

num_threads [an integer] Set number of threads. (Nipype **default** value: 1)

out_file [a pathlike object or string representing a file] Output image file name. Maps to a command-line argument: `-prefix %s`.

outputtype ['NIFTI' or 'AFNI' or 'NIFTI_GZ'] AFNI output filetype.

out_file [a pathlike object or string representing an existing file] Output file.

cmtklib.interfaces.ants module

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

MultipleANTsApplyTransforms

[Link to code](#)

Bases: nipype.interfaces.base.core.BaseInterface

Apply linear and deformable transforms estimated by ANTS to a list of images.

It calls the `antsApplyTransform` on a series of images.

Examples

```
>>> apply_tf = MultipleANTsApplyTransforms()
>>> apply_tf.inputs.input_images = ['/path/to/sub-01_atlas-L2018_desc-scale1_
↳ dseg.nii.gz',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale2_
↳ dseg.nii.gz',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale3_
↳ dseg.nii.gz',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale4_
↳ dseg.nii.gz',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale5_
↳ dseg.nii.gz']
>>> apply_tf.inputs.transforms = ['/path/to/final1Warp.nii.gz',
>>>                                '/path/to/final0GenericAffine.mat']
>>> apply_tf.inputs.reference_image = File(mandatory=True, exists=True)
>>> apply_tf.inputs.interpolation = 'NearestNeighbor'
>>> apply_tf.inputs.default_value = 0.0
>>> apply_tf.inputs.out_postfix = "_transformed"
>>> apply_tf.run()
```

reference_image : a 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'

(Nipype **default** value: Linear)

out_postfix [a string] (Nipype **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 Nipype interfaces for Camino functions missing in nipype or modified.

DTLUTGen

[Link to code](#)

Bases: nipype.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 ≥ 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 range 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: `nipy.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 multi-tensor 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: `nipype.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' 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
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 : 'mrtrix' 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: `nipy.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: `nipy.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] Spherical harmonics order.

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: `nipy.interfaces.dipy.base.DipyDiffusionInterface`

Computes the MAP MRI model.

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: `nipy.interfaces.dipy.base.DipyDiffusionInterface`

Uses SHORE [Merlet2013] to generate the fODF of DWIs.

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

References

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 : 'mrtix' 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: nipype.interfaces.freesurfer.base.FSCommand

Wrapped executable: bregister.

Use FreeSurfer bregister 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.

fsregout_file [a pathlike object or string representing a file] Resulting FSL-Style registration matrix.

regout_file [a pathlike object or string representing a file] Resulting registration file.

copyBrainMaskToFreesurfer

[Link to code](#)

Bases: nipype.interfaces.io.IOBase

Copy a custom brain mask in the freesurfer subject mri/ directory.

It replaces the brainmask files generated by Freesurfer recon-all in order to re-run recon-all with a custom brain mask.

Examples

```
>>> from cmtklib.interfaces.freesurfer import copyBrainMaskToFreesurfer
>>> copy_mask_fs = copyBrainMaskToFreesurfer()
>>> copy_mask_fs.inputs.in_file = 'sub-01_desc-brain_mask.nii.gz'
>>> copy_mask_fs.inputs.subject_dir = '/path/to/output_dir/freesurfer/sub-01'
>>> copy_mask_fs.run()
```

in_file : a pathlike object or string representing an existing file subject_dir : a pathlike object or string representing an existing directory

out_brainmask_file : a pathlike object or string representing an existing file out_brainmaskauto_file : a pathlike object or string representing an existing file

copyFileToFreesurfer

[Link to code](#)

Bases: nipype.interfaces.io.IOBase

Copy a file to an output specified.

Note: Not used.

in_file : a pathlike object or string representing an existing file out_file : a pathlike object or string representing a file

`out_file` : a pathlike object or string representing an existing file

cmtklib.interfaces.fsl module

The FSL module provides Nipype interfaces for FSL functions missing in Nipype or modified.

ApplymultipleWarp

[Link to code](#)

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

Apply a deformation field estimated by FSL `fnirt` to a list of images.

Example

```
>>> from cmtklib.interfaces import fsl
>>> apply_warp = fsl.ApplymultipleWarp()
>>> apply_warp.inputs.in_files = ['/path/to/sub-01_atlas-L2018_desc-scale1_
↳dseg.nii.gz',
>>>                                '/path/to/sub-01_atlas-L2018_desc-scale2_
↳dseg.nii.gz',
>>>                                '/path/to/sub-01_atlas-L2018_desc-scale3_
↳dseg.nii.gz',
>>>                                '/path/to/sub-01_atlas-L2018_desc-scale4_
↳dseg.nii.gz',
>>>                                '/path/to/sub-01_atlas-L2018_desc-scale5_
↳dseg.nii.gz']
>>> apply_warp.inputs.field_file = '/path/to/fnirt_deformation.nii.gz'
>>> apply_warp.inputs.ref_file = '/path/to/sub-01_meanBOLD.nii.gz'
>>> apply_warp.run()
```

field_file [a pathlike object or string representing an existing file] Deformation field.

ref_file [a pathlike object or string representing an existing file] Reference image used for target space.

in_files [a list of items which are a pathlike object or string representing an existing file] Files to be registered.

interp ['nn' or 'trilinear' or 'sinc' or 'spline'] Interpolation method. Maps to a command-line argument: `--interp=%s` (position: -2).

out_files [a list of items which are a pathlike object or string representing a file] Warped files.

ApplymultipleXfm

[Link to code](#)

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

Apply an XFM transform estimated by FSL `flirt` to a list of images.

Example

```
>>> from cmtklib.interfaces import fsl
>>> apply_xfm = fsl.ApplymultipleXfm
>>> apply_xfm.inputs.in_files = ['/path/to/sub-01_atlas-L2018_desc-scale1_dseg.
↪nii.gz',
>>>                               '/path/to/sub-01_atlas-L2018_desc-scale2_dseg.
↪nii.gz',
>>>                               '/path/to/sub-01_atlas-L2018_desc-scale3_dseg.
↪nii.gz',
>>>                               '/path/to/sub-01_atlas-L2018_desc-scale4_dseg.
↪nii.gz',
>>>                               '/path/to/sub-01_atlas-L2018_desc-scale5_dseg.
↪nii.gz']
>>> apply_xfm.inputs.xfm_file = '/path/to/flirt_transform.xfm'
>>> apply_xfm.inputs.reference = '/path/to/sub-01_meanBOLD.nii.gz'
>>> apply_xfm.run()
```

reference [a pathlike object or string representing an existing file] Reference image used for target space.

xfm_file [a pathlike object or string representing an existing file] Transform file.

in_files [a list of items which are a pathlike object or string representing an existing file] Files to be registered.

interp ['nearestneighbour' or 'spline'] Interpolation used.

out_files [a list of items which are a pathlike object or string representing a file] Transformed files.

BinaryThreshold

[Link to code](#)

Bases: `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 cmtklib.interfaces.fsl import CreateAcqpFile
>>> create_acqp = CreateAcqpFile()
>>> create_acqp.inputs.total_readout = 0.28
>>> create_acqp.run()
```

total_readout : a float

acqp : a pathlike object or string representing an existing file

CreateIndexFile

[Link to code](#)

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

Create an index file for FSL eddy from a `mrtrix` diffusion gradient table.

Examples

```
>>> from cmtklib.interfaces.fsl import CreateIndexFile
>>> create_index = CreateIndexFile()
>>> create_index.inputs.in_grad_mrtrix = 'grad.txt'
>>> create_index.run()
```

in_grad_mrtrix [a pathlike object or string representing an existing file] Input DWI gradient table in 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 distorsion correction using FSL eddy.

Example

```
>>> from cmtklib.interfaces import fsl
>>> eddyc = fsl.Eddy(in_file='diffusion.nii',
>>>                  bvecs='diffusion.bvecs',
>>>                  bvals='diffusion.bvals',
>>>                  out_file="diffusion_eddyc.nii")
>>> eddyc.run()
```

acqp [a pathlike object or string representing an existing file] File containing acquisition parameters. Maps to a command-line argument: `--acqp=%s` (position: 3).

bvals [a pathlike object or string representing an existing file] File containing the b-values for all volumes in `-imain`. Maps to a command-line argument: `--bvals=%s` (position: 5).

bvecs [a pathlike object or string representing an existing file] File containing the b-vectors for all volumes in `-imain`. Maps to a command-line argument: `--bvecs=%s` (position: 4).

in_file [a pathlike object or string representing an existing file] File containing all the images to estimate distortions for. Maps to a command-line argument: `--imain=%s` (position: 0).

index [a pathlike object or string representing an existing file] File containing indices for all volumes in `-imain` into `-acqp` and `-topup`. Maps to a command-line argument: `--index=%s` (position: 2).

mask [a pathlike object or string representing an existing file] Mask to indicate brain. Maps to a command-line argument: `--mask=%s` (position: 1).

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

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: {})

out_file [a pathlike object or string representing a file] Basename for output. Maps to a command-line argument: --out=%s (position: 6).

output_type ['NIFTI' or 'NIFTI_PAIR' or 'NIFTI_GZ' or 'NIFTI_PAIR_GZ'] FSL output type.

verbose [a boolean] Display debugging messages. Maps to a command-line argument: --verbose (position: 7).

bvecs_rotated [a pathlike object or string representing an existing file] Path/name of rotated DWI gradient bvecs file.

eddy_corrected [a pathlike object or string representing an existing file] Path/name of 4D eddy corrected DWI file.

EddyOpenMP

[Link to code](#)

Bases: nipype.interfaces.fsl.base.FSLCommand

Wrapped executable: eddy_openmp.

Performs eddy current 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.

cmtklib.interfaces.misc module

ConcatOutputsAsTuple

[Link to code](#)

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

Concatenate 2 different output file as a Tuple of 2 files.

Examples

```
>>> from cmtklib.interfaces.misc import ConcatOutputsAsTuple
>>> concat_outputs = ConcatOutputsAsTuple()
>>> concat_outputs.inputs.input1 = 'output_interface1.nii.gz'
>>> concat_outputs.inputs.input2 = 'output_interface2.nii.gz'
>>> concat_outputs.run()
```

input1 : a pathlike object or string representing an existing file **input2** : a pathlike object or string representing an existing file

out_tuple : a tuple of the form: (a pathlike object or string representing an existing file, a pathlike object or string representing an existing file)

ExtractHeaderVoxel2WorldMatrix

[Link to code](#)

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

Write in a text file the voxel-to-world transform matrix from the header of a Nifti image.

Examples

```
>>> from cmtklib.interfaces.misc import ExtractHeaderVoxel2WorldMatrix
>>> extract_mat = ExtractHeaderVoxel2WorldMatrix()
>>> extract_mat.inputs.in_file = 'sub-01_T1w.nii.gz'
>>> extract_mat.run()
```

in_file [a pathlike object or string representing an existing file] Input image file.

out_matrix [a pathlike object or string representing an existing file] Output voxel to world affine transform file.

ExtractImageVoxelSizes

[Link to code](#)

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

Returns a list of voxel sizes from an image.

Examples

```
>>> from cmtklib.interfaces.misc import ExtractImageVoxelSizes
>>> extract_voxel_sizes = ExtractImageVoxelSizes()
>>> extract_voxel_sizes.inputs.in_file = 'sub-01_T1w.nii.gz'
>>> extract_voxel_sizes.run()
```

`in_file` : a pathlike object or string representing an existing file

`voxel_sizes` : a list of items which are any value

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

Wrapped executable: `dwi2fod`.

Perform non-negativity constrained spherical deconvolution using `dwi2fod`.

Note that this program makes use of implied symmetries in the diffusion profile. First, the fact the signal attenuation profile is real implies that it has conjugate symmetry, i.e. $Y(l, -m) = Y(l, m)^*$ (where $*$ denotes the complex conjugate). Second, the diffusion profile should be antipodally symmetric (i.e. $S(x) = S(-x)$), implying that all odd l components should be zero. Therefore, this program only computes the even elements. Note that the spherical harmonics equations used here differ slightly from those conventionally used, in that the $(-1)^m$ factor has been omitted. This should be taken into account in all subsequent calculations. Each volume in the output image corresponds to a different spherical harmonic component, according to the following convention:

- [0] $Y(0,0)$
- [1] $\text{Im} \{Y(2,2)\}$
- [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] 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/mm²). 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/mm²). See `FSL2MRTrix()`. Maps to a command-line argument: `-grad %s` (position: 2).

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: {})

ignore_slice_by_volume [a list of from 2 to 2 items which are an integer] Requires two values (i.e. [34 1] for [Slice Volume] Ignores the image slices specified when computing the tensor. Slice here means the z coordinate of the slice to be ignored. Maps to a command-line argument: `-ignoreslices %s` (position: 2).

ignore_volumes [a list of at least 1 items which are an integer] Requires two values (i.e. [2 5 6] for [Volumes] Ignores the image volumes specified when computing the tensor. Maps to a command-line argument: `-ignorevolumes %s` (position: 2).

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

out_filename [a pathlike object or string representing a file] Output tensor filename. Maps to a command-line argument: `%s` (position: -1).

quiet [a boolean] Do not display information messages or progress status. Maps to a command-line argument: `-quiet` (position: 1).

tensor [a pathlike object or string representing an existing file] Path/name of output diffusion tensor image.

DWIBiasCorrect

[Link to code](#)

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

Wrapped executable: `dwibiascorrect`.

Correct for bias field in diffusion MRI data using the `dwibiascorrect` tool.

Example

```
>>> from cmtklib.interfaces.mrtrix3 import DWIBiasCorrect
>>> dwi_biascorr = DWIBiasCorrect()
>>> dwi_biascorr.inputs.in_file = 'sub-01_dwi.nii.gz'
>>> dwi_biascorr.inputs.use_ants = True
>>> dwi_biascorr.run()
```

in_file [a pathlike object or string representing an existing file] The input image series to be corrected. Maps to a command-line argument: `%s` (position: -2).

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

debug [a boolean] Display debugging messages. Maps to a command-line argument: `-debug` (position: 5).

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

force_writing [a boolean] Force file overwriting. Maps to a command-line argument: `-force` (position: 4).

mask [a pathlike object or string representing a file] Manually provide a mask image for bias field estimation (optional). Maps to a command-line argument: `-mask %s` (position: 2).

out_bias [a pathlike object or string representing a file] Output the estimated bias field. Maps to a command-line argument: `-bias %s` (position: 3).

out_file [a pathlike object or string representing a file] The output corrected image series. Maps to a command-line argument: `%s` (position: -1).

use_ants [a boolean] Use ANTS N4 to estimate the inhomogeneity field. Maps to a command-line argument: `ants` (position: 1). Mutually **exclusive** with inputs: `use_ants`, `use_fsl`.

use_fsl [a boolean] Use FSL FAST to estimate the inhomogeneity field. Maps to a command-line argument: `fsl` (position: 1). Mutually **exclusive** with inputs: `use_ants`, `use_fsl`.

out_bias [a pathlike object or string representing an existing file] Output estimated bias field.

out_file [a pathlike object or string representing an existing file] Output corrected DWI image.

DWIDenoise

[Link to code](#)

Bases: `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: `nipy.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: `nipy.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/mm²). 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.

ExtractMRTrxGrad

[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.ExtractMRTrxGrad()
>>> mrtrix_grad.inputs.in_file = 'sub-01_dwi.mif'
>>> mrtrix_grad.inputs.out_grad_mrtrix = 'sub-01_gradient.txt'
>>> mrtrix_grad.run()
```

in_file [a pathlike object or string representing an existing file] Input images to be read. Maps to a command-line argument: %s (position: -2).

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

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: {})

out_grad_mrtrix [a pathlike object or string representing a file] Export the DWI gradient table to file in MRtrix format. Maps to a command-line argument: -export_grad_mrtrix %s.

out_grad_mrtrix [a pathlike object or string representing a file] Output MRtrix gradient text file if set.

FilterTractogram

[Link to code](#)

Bases: *MRTrix3Base*

Wrapped executable: `tcksift`.

Spherical-deconvolution informed filtering of tractograms using `tcksift` [Smith2013SIFT].

References

Example

```
>>> import cmtklib.interfaces.mrtrix3 as cmp_mrt
>>> mrtrix_sift = cmp_mrt.FilterTractogram()
>>> mrtrix_sift.inputs.in_tracks = 'tractogram.tck'
>>> mrtrix_sift.inputs.in_fod = 'spherical_harmonics_image.nii.gz'
>>> mrtrix_sift.inputs.out_file = 'sift_tractogram.tck'
>>> mrtrix_sift.run()
```

in_fod [a pathlike object or string representing an existing file] Input image containing the spherical harmonics of the fibre orientation distributions. Maps to a command-line argument: `%s` (position: -2).

in_tracks [a pathlike object or string representing an existing file] Input track file in TCK format. Maps to a command-line argument: `%s` (position: -3).

act_file [a pathlike object or string representing an existing file] ACT 5TT image file. Maps to a command-line argument: `-act %s` (position: -4).

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

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

out_file [a pathlike object or string representing a file] Output filtered tractogram. Maps to a command-line argument: `%s` (position: -1).

out_tracks [a pathlike object or string representing an existing file] Output filtered tractogram.

Generate5tt

[Link to code](#)

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

Wrapped executable: `5ttgen`.

Generate a 5TT image suitable for ACT using the selected algorithm using `5ttgen`.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> gen5tt = mrt.Generate5tt()
>>> gen5tt.inputs.in_file = 'T1.nii.gz'
>>> gen5tt.inputs.algorithm = 'fsl'
>>> gen5tt.inputs.out_file = '5tt.mif'
>>> gen5tt.cmdline
'5ttgen fsl T1.nii.gz 5tt.mif'
>>> gen5tt.run()
```

algorithm ['fsl' or 'gif' or 'freesurfer' or 'hsvs'] Tissue segmentation algorithm. Maps to a command-line argument: %s (position: -3).

in_file [a pathlike object or string representing an existing file] Input image. Maps to a command-line argument: -nocrop -sgm_amyg_hipp %s (position: -2).

out_file [a pathlike object or string representing a file] Output image. Maps to a command-line argument: %s (position: -1).

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

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: {})

out_file [a pathlike object or string representing an existing file] Output image.

GenerateGMWMInterface

[Link to code](#)

Bases: nipype.interfaces.base.core.CommandLine

Wrapped executable: 5tt2gmwmi.

Generate a grey matter-white matter interface mask from the 5TT image using 5tt2gmwmi.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as cmp_mrt
>>> genWMGMI = cmp_mrt.Generate5tt()
>>> genWMGMI.inputs.in_file = '5tt.mif'
>>> genWMGMI.inputs.out_file = 'gmwmi.mif'
>>> genGMWMI.run()
```

in_file [a pathlike object or string representing an existing file] Input 5TT image. Maps to a command-line argument: %s (position: -2).

out_file [a pathlike object or string representing a file] Output GW/WM interface image. Maps to a command-line argument: %s (position: -1).

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

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: {})

out_file [a pathlike object or string representing an existing file] Output image.

MRConvert

[Link to code](#)

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

Wrapped executable: `mrconvert`.

Perform conversion with `mrconvert` between different file types and optionally extract a subset of the input image.

If used correctly, this program can be a very useful workhorse. In addition to converting images between different formats, it can be used to extract specific studies from a data set, extract a specific region of interest, flip the images, or to scale the intensity of the images.

Example

```
>>> import cmtklib.interfaces.mrtrix3 as mrt
>>> mrconvert = mrt.MRConvert()
>>> mrconvert.inputs.in_file = 'dwi_FA.mif'
>>> mrconvert.inputs.out_filename = 'dwi_FA.nii'
>>> mrconvert.run()
```

in_dir [a pathlike object or string representing an existing directory] Directory containing DICOM files. Maps to a command-line argument: `%s` (position: -2). Mutually **exclusive** with inputs: `in_file`, `in_dir`.

in_file [a pathlike object or string representing an existing file] Voxel-order data filename. Maps to a command-line argument: `%s` (position: -2). Mutually **exclusive** with inputs: `in_file`, `in_dir`.

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

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

extension ['mif' or 'nii' or 'float' or 'char' or 'short' or 'int' or 'long' or 'double'] "i.e. Bfloat". Can be "char", "short", "int", "long", "float" or "double". (Nipype **default** value: `mif`)

extract_at_axis [1 or 2 or 3] Extract data only at the coordinates specified. This option specifies the Axis. Must be used in conjunction with `extract_at_coordinate`. . Maps to a command-line argument: `-coord %s` (position: 1).

extract_at_coordinate [a list of from 1 to 3 items which are an integer] Extract data only at the coordinates specified. This option specifies the coordinates. Must be used in conjunction with `extract_at_axis`. Three comma-separated numbers giving the size of each voxel in mm. Maps to a command-line argument: `%s` (position: 2).

force_writing [a boolean] Force file overwriting. Maps to a command-line argument: `-force`.

grad [a pathlike object or string representing an existing file] Gradient encoding, supplied as a 4xN text file with each line is in the format [X Y Z b], where [X Y Z] describe the direction of the applied gradient, and b gives the b-value in units (1000 s/mm^2). See FSL2MRTrix. Maps to a command-line argument: `-grad %s` (position: 9).

grad_fsl [a tuple of the form: (a pathlike object or string representing an existing file, a pathlike object or string representing an existing file)] [bvecs, bvals] DW gradient scheme (FSL format). Maps to a command-line argument: `-fslgrad %s %s`.

- layout** ['nii' or 'float' or 'char' or 'short' or 'int' or 'long' or 'double'] Specify the layout of the data in memory. The actual layout produced will depend on whether the output image format can support it. Maps to a command-line argument: `-output %s` (position: 5).
- offset_bias** [a float] Apply offset to the intensity values. Maps to a command-line argument: `-scale %d` (position: 7).
- out_filename** [a pathlike object or string representing a file] Output filename. Maps to a command-line argument: `%s` (position: -1).
- output_datatype** ['float32' or 'float32le' or 'float32be' or 'float64' or 'float64le' or 'float64be' or 'int64' or 'uint64' or 'int64le' or 'uint64le' or 'int64be' or 'uint64be' or 'int32' or 'uint32' or 'int32le' or 'uint32le' or 'int32be' or 'uint32be' or 'int16' or 'uint16' or 'int16le' or 'uint16le' or 'int16be' or 'uint16be' or 'cfloat32' or 'cfloat32le' or 'cfloat32be' or 'cfloat64' or 'cfloat64le' or 'cfloat64be' or 'int8' or 'uint8' or 'bit'] Specify output image data type. Valid choices are: float32, float32le, float32be, float64, float64le, float64be, int64, uint64, int64le, uint64le, int64be, uint64be, int32, uint32, int32le, uint32le, int32be, uint32be, int16, uint16, int16le, uint16le, int16be, uint16be, cfloat32, cfloat32le, cfloat32be, cfloat64, cfloat64le, cfloat64be, int8, uint8, bit.”. Maps to a command-line argument: `-datatype %s` (position: 2).
- prs** [a boolean] Assume that the DW gradients are specified in the PRS frame (Siemens DICOM only). Maps to a command-line argument: `-prs` (position: 3).
- quiet** [a boolean] Do not display information messages or progress status. Maps to a command-line argument: `-quiet`.
- replace_nan_with_zero** [a boolean] Replace all NaN values with zero. Maps to a command-line argument: `-zero` (position: 8).
- resample** [a float] Apply scaling to the intensity values. Maps to a command-line argument: `-scale %d` (position: 6).
- stride** [a list of from 3 to 4 items which are an integer] Three to four comma-separated numbers specifying the strides of the output data in memory. The actual strides produced will depend on whether the output image format can support it.. Maps to a command-line argument: `-stride %s` (position: 3).
- voxel_dims** [a list of from 3 to 3 items which are a float] Three comma-separated numbers giving the size of each voxel in mm. Maps to a command-line argument: `-vox %s` (position: 3).
- converted** [a pathlike object or string representing an existing file] Path/name of 4D volume in voxel order.

MRCrop

[Link to code](#)

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

Wrapped executable: `mrcalc`.

Multiply two images together using `mrcalc` tool.

Examples

```
>>> from cmtklib.interfaces.mrtrix3 import MRtrix_mul
>>> multiply = MRtrix_mul()
>>> multiply.inputs.input1 = 'image1.nii.gz'
>>> multiply.inputs.input2 = 'image2.nii.gz'
>>> multiply.inputs.out_filename = 'result.nii.gz'
>>> multiply.run()
```

input1 [a pathlike object or string representing an existing file] Input1 file. Maps to a command-line argument: `%s` (position: 1).

input2 [a pathlike object or string representing an existing file] Input2 file. Maps to a command-line argument: `%s` (position: 2).

out_filename [a string] Out filename. Maps to a command-line argument: `-mult %s` (position: 3).

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

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: {})

out_file [a pathlike object or string representing a file] Multiplication result file.

SIFT2

[Link to code](#)

Bases: *MRTrix3Base*

Wrapped executable: `tcksift2`.

Determine an appropriate cross-sectional area multiplier for each streamline using `tcksift2` [Smith2015SIFT2].

References

Example

```
>>> import cmtklib.interfaces.mrtrix3 as cmp_mrt
>>> mrtrix_sift2 = cmp_mrt.SIFT2()
>>> mrtrix_sift2.inputs.in_tracks = 'tractogram.tck'
>>> mrtrix_sift2.inputs.in_fod = 'spherical_harmonics_image.nii.gz'
>>> mrtrix_sift2.inputs.out_file = 'sift2_fiber_weights.txt'
>>> mrtrix_sift2.run()
```

in_fod [a pathlike object or string representing an existing file] Input image containing the spherical harmonics of the fibre orientation distributions. Maps to a command-line argument: %s (position: -2).

in_tracks [a pathlike object or string representing an existing file] Input track file in TCK format. Maps to a command-line argument: %s (position: -3).

act_file [a pathlike object or string representing an existing file] ACT 5TT image file. Maps to a command-line argument: -act %s (position: -4).

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

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: {})

out_file [a pathlike object or string representing a file] Output filtered tractogram. Maps to a command-line argument: %s (position: -1).

out_tracks [a pathlike object or string representing an existing file] Output filtered tractogram.

StreamlineTrack

[Link to code](#)

Bases: `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 preceding 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_gmwm [a boolean] Crop streamline endpoints more precisely as they cross the GM-WM interface. Maps to a command-line argument: `-crop_at_gmwm`.

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: `-noprocomputed`.

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/mm²). 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 cmtklib.interfaces.mrtrix3 as mrt
>>> tensor2vector = mrt.Tensor2Vector()
>>> tensor2vector.inputs.in_file = 'dwi_tensor.mif'
>>> tensor2vector.run()
```

in_file [a pathlike object or string representing an existing file] Diffusion tensor image. Maps to a command-line argument: `%s` (position: -2).

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

debug [a boolean] Display debugging messages. Maps to a command-line argument: `-debug` (position: 1).

environ [a dictionary with keys which are a bytes or None or a value of class 'str' and with values which are a bytes or None or a value of class 'str'] Environment variables. (Nipype **default** value: `{}`)

out_filename [a pathlike object or string representing a file] Output vector filename. Maps to a command-line argument: `-vector %s` (position: -1).

quiet [a boolean] Do not display information messages or progress status. Maps to a command-line argument: `-quiet` (position: 1).

vector [a pathlike object or string representing an existing file] The output image of the major eigenvectors of the diffusion tensor image.

Submodules

cmtklib.config module

Module that defines methods for handling CMP3 configuration files.

`cmtklib.config.anat_load_config_json(pipeline, config_path)`

Load the JSON configuration file of an anatomical pipeline.

Parameters

- **pipeline** (`Instance(cmp.pipelines.anatomical.anatomical.AnatomicalPipeline)`) – Instance of AnatomicalPipeline
- **config_path** (`string`) – Path of the JSON configuration file

`cmtklib.config.anat_save_config(pipeline, config_path)`

Save the configuration file of an anatomical pipeline.

Parameters

- **pipeline** (`Instance(cmp.pipelines.anatomical.anatomical.AnatomicalPipeline)`) – Instance of AnatomicalPipeline

- **config_path** (*string*) – Path of the JSON configuration file

`cmtklib.config.check_configuration_format(config_path)`

Check format of the configuration file.

Parameters **config_path** (*string*) – Path to pipeline configuration file

Returns **ext** – Format extension of the pipeline configuration file

Return type 'ini' or 'json'

`cmtklib.config.check_configuration_version(config)`

Check the version of CMP3 used to generate a configuration.

Parameters **config** (*Dict*) – Dictionary of configuration parameters loaded from JSON file

Returns **is_same** – **True** if the version used to generate the configuration matches the version currently used (`cmp.info.__version__`).

Return type **bool**

`cmtklib.config.convert_config_ini_2_json(config_ini_path)`

Convert a configuration file in old INI format to new JSON format.

Parameters **config_ini_path** (*string*) – Path to configuration file in old INI format

Returns **config_json_path** – Path to converted configuration file in new JSON format

Return type **string**

`cmtklib.config.create_configparser_from_pipeline(pipeline, debug=False)`

Create a ConfigParser object from a Pipeline instance.

Parameters

- **pipeline** (*Instance(Pipeline)*) – Instance of pipeline
- **debug** (*bool*) – If **True**, show additional prints

Returns **config** – Instance of ConfigParser

Return type *Instance(configparser.ConfigParser)*

`cmtklib.config.create_subject_configuration_from_ref(project, ref_conf_file, pipeline_type, multiproc_number_of_cores=1)`

Create the pipeline configuration file for an individual subject from a reference given as input.

Parameters

- **project** (*cmp.project.CMP_Project_Info*) – Instance of *cmp.project.CMP_Project_Info*
- **ref_conf_file** (*string*) – Reference configuration file
- **pipeline_type** (*'anatomical', 'diffusion', 'fMRI'*) – Type of pipeline
- **multiproc_number_of_cores** (*int*) – Number of threads used by Nipype

Returns **subject_conf_file** – Configuration file of the individual subject

Return type **string**

`cmtklib.config.dmri_load_config_json(pipeline, config_path)`

Load the JSON configuration file of a diffusion pipeline.

Parameters

- **pipeline** (*Instance(cmp.pipelines.diffusion.diffusion.DiffusionPipeline)*) – Instance of DiffusionPipeline
- **config_path** (*string*) – Path of the JSON configuration file

`cmtklib.config.dmri_save_config(pipeline, config_path)`

Save the INI configuration file of a diffusion pipeline.

Parameters

- **pipeline** (*Instance(cmp.pipelines.diffusion.diffusion.DiffusionPipeline)*) – Instance of DiffusionPipeline
- **config_path** (*string*) – Path of the JSON configuration file

`cmtklib.config.fmri_load_config_json(pipeline, config_path)`

Load the JSON configuration file of a fMRI pipeline.

Parameters

- **pipeline** (*Instance(cmp.pipelines.functional.fMRI.fMRIPipeline)*) – Instance of fMRIPipeline
- **config_path** (*string*) – Path of the JSON configuration file

`cmtklib.config.fmri_save_config(pipeline, config_path)`

Save the INI configuration file of a fMRI pipeline.

Parameters

- **pipeline** (*Instance(cmp.pipelines.functional.fMRI.fMRIPipeline)*) – Instance of fMRIPipeline
- **config_path** (*string*) – Path of the JSON configuration file

`cmtklib.config.get_anat_process_detail_json(project_info, section, detail)`

Get the value for a parameter key (detail) in the stage section of the anatomical JSON config file.

Parameters

- **project_info** (*Instance(cmp.project.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_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_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.config.save_configparser_as_json(config, config_json_path, ini_mode=False, debug=False)`

Save a ConfigParser to JSON file.

Parameters

- **config** (*Instance(configparser.ConfigParser)*) – Instance of ConfigParser
- **config_json_path** (*string*) – Output path of JSON configuration file
- **ini_mode** (*bool*) – If `True`, handles all content stored in strings
- **debug** (*bool*) – If `True`, show additional prints

`cmtklib.config.set_pipeline_attributes_from_config(pipeline, config, debug=False)`

Set the pipeline stage attributes given a configuration.

Parameters

- **pipeline** (*Instance(Pipeline)*) – Instance of pipeline
- **config** (*Dict*) – Dictionary of configuration parameter loaded from the JSON configuration file
- **debug** (*bool*) – If `True`, show additional prints

cmtklib.connectome module

Module that defines CMTK functions and Nipype interfaces for connectome mapping.

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 ['Lausanne2018' or 'NativeFreesurfer' or 'Custom'] Parcellation scheme. (Nipype default value: Lausanne2018)

roi_graphmls [a list of items which are a pathlike object or string representing an existing file] GraphML description of ROI volumes (Lausanne2018).

roi_volumes [a list of items which are a pathlike object or string representing an existing file] ROI volumes registered to diffusion space.

voxel_connectivity [a list of items which are a pathlike object or string representing an existing file] ProbtrackX connectivity matrices (# seed voxels x # target ROIs).

connectivity_matrices [a list of items which are a pathlike object or string representing a file] Connectivity matrices.

endpoints_file [a pathlike object or string representing a file] Numpy files storing the list of fiber endpoint.

endpoints_mm_file [a pathlike object or string representing a file] Numpy files storing the list of fiber endpoint in mm.

filtered_fiberslabel_files [a list of items which are a pathlike object or string representing a file] List of fiber start end ROI parcellation label after filtering.

final_fiberlabels_files [a list of items which are a pathlike object or string representing a file] List of fiber start end ROI parcellation label.

final_fiberslength_files [a list of items which are a pathlike object or string representing a file] List of fiber length.

streamline_final_file [a pathlike object or string representing a file] Final tractogram of fibers considered in the creation of connectivity matrices.

CMTK_rsfmri_cmat

[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 cmtklib.connectome import CMTK_rsfmri_cmat
>>> cmat = CMTK_rsfmri_cmat()
>>> 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',
```

(continues on next page)

(continued from previous page)

```

>>>                                     '/path/to/sub-01_space-meanBOLD_atlas-L2018_
↳desc-scale5_dseg.nii.gz']
>>> cmat.inputs.roi_graphmls = ['/path/to/sub-01_atlas-L2018_desc-scale1_dseg.
↳graphml',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale2_dseg.
↳graphml',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale3_dseg.
↳graphml',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale4_dseg.
↳graphml',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale5_dseg.
↳graphml']
>>> cmat.inputs.parcellation_scheme = 'Lausanne2018'
>>> cmat.inputs.apply_scrubbing = False
>>> cmat.inputs.output_types = ['gPickle', 'mat', 'graphml']
>>> cmat.run()

```

func_file [a pathlike object or string representing an existing file] FMRI volume.

DVARS [a pathlike object or string representing an existing file] DVARS file if scrubbing is performed.

DVARS_th [a float] DVARS threshold.

FD [a pathlike object or string representing an existing file] FD file if scrubbing is performed.

FD_th [a float] FD threshold.

apply_scrubbing [a boolean] Apply scrubbing.

atlas_info [a dictionary with keys which are any value and with values which are any value] Custom atlas information.

output_types [a list of items which are a string] Output types of the connectivity matrices.

parcellation_scheme ['Lausanne2018' or 'NativeFreesurfer' or 'Custom'] Parcellation scheme. (Nipype default value: Lausanne2018)

roi_graphmls [a list of items which are a pathlike object or string representing an existing file] GraphML description file for ROI volumes (used only if parcellation_scheme == Lausanne2018).

roi_volumes [a list of items which are a pathlike object or string representing an existing file] ROI volumes registered to functional space.

avg_timeseries [a list of items which are a pathlike object or string representing an existing file] ROI average timeseries.

connectivity_matrices [a list of items which are a pathlike object or string representing an existing file] Functional connectivity matrices.

scrubbed_idx [a pathlike object or string representing an existing file] Scrubbed indices.

`cmtklib.connectome.cmat(intrk, roi_volumes=None, roi_graphmls=None, parcellation_scheme=None, compute_curvature=True, additional_maps=None, output_types=None, atlas_info=None)`

Create the connection matrix for each resolution using fibers and ROIs.

Parameters

- **intrk** (*TRK file*) – Reconstructed tractogram
- **roi_volumes** (*list*) – List of parcellation files for a given parcellation scheme

- **roi_graphmls** (*list*) – List of graphmls files that describes parcellation nodes
- **parcellation_scheme** (*['NativeFreesurfer', 'Lausanne2018', 'Custom']*) –
- **compute_curvature** (*Boolean*) –
- **additional_maps** (*dict*) – A dictionary of key/value for each additional map where the value is the path to the map
- **output_types** (*['gPickle', 'mat', 'graphml']*) –
- **atlas_info** (*dict*) – Dictionary storing information such as path to files related to a parcellation atlas / scheme.

`cmtklib.connectome.compute_curvature_array(fib)`

Computes the curvature array.

`cmtklib.connectome.create_endpoints_array(fib, voxelSize, print_info)`

Create the endpoints arrays for each fiber.

Parameters

- **fib** (*the fibers data*) –
- **voxelSize** (*3-tuple*) – It contains the voxel size of the ROI image
- **print_info** (*bool*) – If True, print extra information

Returns

- **(endpoints** (*matrix of size [#fibers, 2, 3] containing for each fiber the*) – index of its first and last point in the voxelSize volume
- **endpointsmm**) (*endpoints in milimeter coordinates*)

`cmtklib.connectome.group_analysis_sconn(output_dir, subjects_to_be_analyzed)`

Perform group level analysis of structural connectivity matrices.

`cmtklib.connectome.save_fibers(oldhdr, oldfib, fname, indices)`

Stores a new trackvis file fname using only given indices.

Parameters

- **oldhdr** (*the tractogram header*) – Tractogram header to use as reference
- **oldfib** (*the fibers data*) – Input fibers
- **fname** (*string*) – Output tractogram filename
- **indices** (*list*) – Indices of fibers included

cmtklib.diffusion module

Module that defines CMTK utility functions for the diffusion pipeline.

ExtractPVEsFrom5TT

[Link to code](#)

Bases: `nipy.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 flipping 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: `nipy.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: `nipy.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: `nipy.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: `nipy.interfaces.base.core.BaseInterface`

Convert a tractogram in `mrtrix` TCK format to `trackvis` TRK format.

Examples

```
>>> from cmtklib.diffusion import Tck2Trk
>>> tck_to_trk = Tck2Trk()
>>> tck_to_trk.inputs.in_tracks = 'sub-01_tractogram.tck'
>>> tck_to_trk.inputs.in_image = 'sub-01_desc-preproc_dwi.nii.gz'
>>> tck_to_trk.inputs.out_tracks = 'sub-01_tractogram.trk'
>>> tck_to_trk.run()
```

in_image [a pathlike object or string representing an existing file] Input image used to extract the header.

in_tracks [a pathlike object or string representing an existing file] Input track file in MRtrix .tck format.

out_tracks [a pathlike object or string representing a file] Output track file in Trackvis .trk format.

out_tracks [a pathlike object or string representing an existing file] Output track file in Trackvis .trk format.

UpdateGMWMInterfaceSeeding

[Link to code](#)

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

Add extra Lausanne2018 structures to the Gray-matter/White-matter interface for tractography seeding.

Examples

```
>>> from cmtklib.diffusion import UpdateGMWMInterfaceSeeding
>>> update_gmwmi = UpdateGMWMInterfaceSeeding()
>>> update_gmwmi.inputs.in_gmwmi_file = 'sub-01_label-gmwmi_desc-orig_dseg.nii.
↪gz'
>>> update_gmwmi.inputs.out_gmwmi_file = 'sub-01_label-gmwmi_desc-modif_dseg.
↪nii.gz'
>>> update_gmwmi.inputs.in_roi_volumes = ['sub-01_space-DWI_atlas-L2018_desc-
↪scale1_dseg.nii.gz',
>>>                                     'sub-01_space-DWI_atlas-L2018_desc-
↪scale2_dseg.nii.gz',
>>>                                     'sub-01_space-DWI_atlas-L2018_desc-
↪scale3_dseg.nii.gz',
>>>                                     'sub-01_space-DWI_atlas-L2018_desc-
↪scale4_dseg.nii.gz',
>>>                                     'sub-01_space-DWI_atlas-L2018_desc-
↪scale5_dseg.nii.gz']
>>> update_gmwmi.run()
```

in_gmwmi_file [a pathlike object or string representing an existing file] Input GMWM interface image used for streamline seeding.

in_roi_volumes [a list of items which are a pathlike object or string representing an existing file] Input parcellation images.

out_gmwmi_file [a pathlike object or string representing a file] Output GM WM interface used for streamline seeding.

out_gmwmi_file [a pathlike object or string representing an existing file] Output GM WM interface used for streamline seeding.

`cmtklib.diffusion.compute_length_array(trkfile=None, streams=None, savefname='lengths.npy')`

Computes the length of the fibers in a tractogram and returns an array of length.

Parameters

- **trkfile** (*TRK file*) – Path to the tractogram in TRK format
- **streams** (*the fibers data*) – The fibers from which we want to compute the length
- **savefname** (*string*) – Output filename to write the length array

Returns `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',
```

(continues on next page)

(continued from previous page)

```
>>>                                     '/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']
>>> 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: `nipy.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
```

(continues on next page)

(continued from previous page)

```
>>> parc_combine.inputs.subjects_dir = '/path/to/output_dir/freesurfer')
>>> parc_combine.inputs.subject_id = 'sub-01'
>>> parc_combine.run()
```

brainstem_structures [a pathlike object or string representing a file] Brainstem segmentation file.

create_colorLUT [a boolean] If **True**, create the color lookup table in Freesurfer format.

create_graphml [a boolean] If **True**, create the parcellation node description files in graphml format.

input_rois [a list of items which are a pathlike object or string representing an existing file] Input parcellation files.

lh_hippocampal_subfields [a pathlike object or string representing a file] Input hippocampal subfields file for left hemisphere.

rh_hippocampal_subfields [a pathlike object or string representing a file] Input hippocampal subfields file for right hemisphere.

subject_id [a string] Freesurfer subject id.

subjects_dir [a pathlike object or string representing a directory] Freesurfer subjects dir.

thalamus_nuclei [a pathlike object or string representing a file] Thalamic nuclei segmentation file.

verbose_level [1 or 2] Verbose level (1: partial (default) / 2: full).

aparc_aseg [a pathlike object or string representing a file] Modified Freesurfer aparc+aseg file.

colorLUT_files [a list of items which are a pathlike object or string representing an existing file] Color lookup table files in Freesurfer format.

graphML_files [a list of items which are a pathlike object or string representing an existing file] Parcellation node description files in graphml format.

output_rois [a list of items which are a pathlike object or string representing an existing file] Output parcellation with all structures combined.

`CombineParcellations.ismember(b)`

ComputeParcellationRoiVolumes

[Link to code](#)

Bases: `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',
```

(continues on next page)

(continued from previous page)

```

>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale5_
↳dseg.nii.gz']
>>> compute_vol.inputs.roi_graphmls = ['/path/to/sub-01_atlas-L2018_desc-
↳scale1_dseg.graphml',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale2_dseg.
↳graphml',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale3_dseg.
↳graphml',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale4_dseg.
↳graphml',
>>>                                     '/path/to/sub-01_atlas-L2018_desc-scale5_dseg.
↳graphml']
>>> compute_vol.inputs.parcellation_scheme = ['Lausanne2018']
>>> compute_vol.run()

```

parcellation_scheme ['NativeFreesurfer' or 'Lausanne2018' or 'Custom'] Parcellation scheme. (Nipype **default** value: Lausanne2018)

roi_graphMLs [a list of items which are a pathlike object or string representing an existing file] GraphML description of ROI volumes (Lausanne2018).

roi_volumes [a list of items which are a pathlike object or string representing an existing file] ROI volumes registered to diffusion space.

roi_volumes_stats [a list of items which are a pathlike object or string representing a file] TSV files with computed parcellation ROI volumes.

Parcellate

[Link to code](#)

Bases: nipype.interfaces.base.core.BaseInterface

Subdivides segmented ROI file into smaller subregions.

This interface interfaces with the CMTK parcellation functions available in `cmtklib.parcellation` module for all parcellation resolutions of a given scheme.

Example

```

>>> from cmtklib.parcellation import Parcellate
>>> parcellate = Parcellate()
>>> parcellate.inputs.subjects_dir = '/path/to/output_dir/freesurfer'
>>> parcellate.inputs.subject_id = 'sub-01'
>>> parcellate.inputs.parcellation_scheme = 'Lausanne2018'
>>> parcellate.run()

```

subject_id [a string] Subject ID.

erode_masks [a boolean] If `True` erode the masks.

parcellation_scheme ['Lausanne2018' or 'NativeFreesurfer'] Parcellation scheme. (Nipype **default** value: Lausanne2018)

subjects_dir [a pathlike object or string representing a directory] Freesurfer main directory.

T1 [a pathlike object or string representing a file] T1 image file.

aparc_aseg [a pathlike object or string representing a file] APArc+ASeg image file (in native space).

aseg [a pathlike object or string representing a file] ASeg image file (in native space).

brain [a pathlike object or string representing a file] Brain-masked T1 image file.

brain_eroded [a pathlike object or string representing a file] Eroded brain file in original space.

brain_mask [a pathlike object or string representing a file] Brain mask file.

csf_eroded [a pathlike object or string representing a file] Eroded csf file in original space.

csf_mask_file [a pathlike object or string representing a file] Cerebrospinal fluid (CSF) mask file.

gray_matter_mask_file [a pathlike object or string representing a file] Cortical gray matter (GM) mask file.

ribbon_file [a pathlike object or string representing an existing file] Image file detailing the cortical ribbon.

roi_files_in_structural_space [a list of items which are a pathlike object or string representing an existing file] ROI image resliced to the dimensions of the original structural image.

white_matter_mask_file [a pathlike object or string representing a file] White matter (WM) mask file.

wm_eroded [a pathlike object or string representing a file] Eroded wm file in original space.

ParcellateBrainstemStructures

[Link to code](#)

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

Parcellates the hippocampal subfields using Freesurfer [Iglesias2015Hippo].

References

Examples

```
>>> parc_hippo = ParcellateHippocampalSubfields()
>>> parc_hippo.inputs.subjects_dir = '/path/to/derivatives/freesurfer'
>>> parc_hippo.inputs.subject_id = 'sub-01'
>>> parc_hippo.run()
```

subject_id [a string] Subject ID.

subjects_dir [a pathlike object or string representing a directory] Freesurfer main directory.

lh_hipposubfields [a pathlike object or string representing a file] Left hemisphere hippocampal subfields file.

rh_hipposubfields [a pathlike object or string representing a file] Right hemisphere hippocampal subfields file.

ParcellateThalamus

[Link to code](#)

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

Parcellates the thalamus into 8 nuclei using an atlas-based method [Najdenovska18].

References

Examples

```
>>> parc_thal = ParcellateThalamus()
>>> parc_thal.inputs.T1w_image = File(mandatory=True, desc='T1w image to be
↳parcellated')
>>> parc_thal.inputs.bids_dir = Directory(desc='BIDS root directory')
>>> parc_thal.inputs.subject = '01'
>>> parc_thal.inputs.template_image = '/path/to/atlas/T1w.nii.gz'
>>> parc_thal.inputs.thalamic_nuclei_maps = '/path/to/atlas/nuclei/probability/
↳map.nii.gz'
>>> parc_thal.inputs.subjects_dir = '/path/to/output_dir/freesurfer'
>>> parc_thal.inputs.subject_id = 'sub-01'
>>> parc_thal.inputs.ants_precision_type = 'float'
>>> parc_thal.run()
```

T1w_image [a pathlike object or string representing a file] T1w image to be parcellated.

subject_id [a string] Subject ID.

subjects_dir [a pathlike object or string representing a directory] Freesurfer main directory.

template_image [a pathlike object or string representing a file] Template T1w.

thalamic_nuclei_maps [a pathlike object or string representing a file] Probability maps of thalamic nuclei (4D image) in template space.

ants_precision_type ['double' or 'float'] Precision type used during computation.

bids_dir [a pathlike object or string representing a directory] BIDS root directory.

session [a string] Session id.

subject [a string] Subject id.

inverse_warped_image [a pathlike object or string representing a file] Inverse warped template.

max_prob_registered [a pathlike object or string representing a file] Max probability label image (native).

prob_maps_registered [a pathlike object or string representing a file] Probabilistic map of thalamus nuclei (native).

thalamus_mask [a pathlike object or string representing a file] Thalamus mask.

transform_file [a pathlike object or string representing a file] Transform file.

warp_file [a pathlike object or string representing a file] Deformation file.

warped_image [a pathlike object or string representing a file] Template registered to T1w image (native).

`cmtklib.parcellation.create_T1_and_Brain(subject_id, subjects_dir)`

Generates T1, T1 masked and aseg+aparc Freesurfer images in NIFTI format.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output_dir/freesurfer)

`cmtklib.parcellation.create_roi(subject_id, subjects_dir, v=True)`

Iteratively creates the ROI_%s.nii.gz files using the given Lausanne2018 parcellation information from networks.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output_dir/freesurfer)
- **v** (*Boolean*) – Verbose mode

`cmtklib.parcellation.create_wm_mask(subject_id, subjects_dir, v=True)`

Creates the white-matter mask using the Freesurfer ribbon as basis in the Lausanne2018 framework.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output_dir/freesurfer)
- **v** (*Boolean*) – Verbose mode

`cmtklib.parcellation.crop_and_move_WM_and_GM(subject_id, subjects_dir)`

Convert Freesurfer images back to original native space when NativeFreesurfer parcellation scheme is used.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output_dir/freesurfer)

`cmtklib.parcellation.crop_and_move_datasets(subject_id, subjects_dir)`

Convert Freesurfer images back to original native space when Lausanne2018 parcellation schemes are used.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output_dir/freesurfer)

`cmtklib.parcellation.erode_mask(fkdir, mask_file)`

Erodes the mask and saves it the Freesurfer subject directory.

Parameters

- **fsdir** (*string*) – Freesurfer subject directory
- **mask_file** (*string*) – Path to mask file

`cmtklib.parcellation.extract(Z, shape, position, fill)`

Extract voxel neighbourhood.

Parameters

- **Z** (*numpy.array*) – The original data
- **shape** (*tuple*) – Tuple containing neighbourhood dimensions
- **position** (*tuple*) – Tuple containing central point indexes
- **fill** (*value*) – Value for the padding of Z

Returns **R** – The output neighbourhood of the specified point in Z

Return type `numpy.array`

`cmtklib.parcellation.generate_WM_and_GM_mask(subject_id, subjects_dir)`

Generates the white-matter and gray-matter masks when NativeFreesurfer parcellation is used.

Parameters

- **subject_id** (*string*) – Freesurfer subject id
- **subjects_dir** (*string*) – Freesurfer subjects dir (Typically /path/to/output_dir/freesurfer)

`cmtklib.parcellation.get_parcellation(parcel='NativeFreesurfer')`

Returns a dictionary containing atlas information.

Note: `atlas_info` often used in the code refers to such a dictionary.

Parameters **parcel** (*parcellation scheme*) – It can be: ‘NativeFreesurfer’ or ‘Lausanne2018’

cmtklib.util module

Module that defines CMTK Utility functions.

class `cmtklib.util.BColors`

Bases: `object`

Utility class for color unicode.

BOLD = `'\x1b[1m'`

ENDC = `'\x1b[0m'`

FAIL = `'\x1b[91m'`

HEADER = `'\x1b[95m'`

OKBLUE = `'\x1b[94m'`

OKGREEN = `'\x1b[92m'`

UNDERLINE = `'\x1b[4m'`

WARNING = `'\x1b[93m'`

`cmtklib.util.check_directory_exists(mandatory_dir)`

Makes sure the mandatory directory exists.

Raises `FileNotFoundError` – Raised when the directory is not found.

`cmtklib.util.convertList2Tuple(lists)`

Convert list of files to tuple of files.

(Duplicated with preprocessing, could be moved to utils in the future)

Parameters `lists` (`[bvecs, bvals]`) – List of files containing bvecs and bvals

Returns `out_tuple` – Tuple of files containing bvecs and bvals

Return type (`bvecs, bvals`)

`cmtklib.util.extract_freesurfer_subject_dir(reconall_report, local_output_dir=None, debug=False)`

Extract Freesurfer subject directory from the report created by Nipype Freesurfer Recon-all node.

Parameters

- **reconall_report** (`string`) – Path to the recon-all report
- **local_output_dir** (`string`) – Local output / derivatives directory
- **debug** (`bool`) – If `True`, show printed outputs

Returns `fs_subject_dir` – Freesurfer subject directory

Return type `string`

`cmtklib.util.extract_reconall_base_dir(file)`

Extract Recon-all base directory from a file.

Parameters `file` (`File`) – File generated by Recon-all

Returns `out_path` – Recon-all base directory

Return type `string`

`cmtklib.util.get_basename(path)`

Return `os.path.basename()` of a path.

Parameters `path` (`os.path`) – Path to extract the containing directory

Returns `path` – Path to the containing directory

Return type `os.path`

`cmtklib.util.get_freesurfer_subject_id(file)`

Extract Freesurfer subject ID from file generated by recon-all.

Parameters `file` (*str*) – File generated by recon-all

Returns `out` – Freesurfer subject ID

Return type *str*

`cmtklib.util.get_node_dictionary_outputs(node_report, local_output_dir=None, debug=False)`

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
- `debug` (*bool*) – If `True`, print output dictionary

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, debug=False)`

Read the Nipype datasink report and return a dictionary of pipeline outputs.

Parameters

- `datasink_report` (*string*) – Path to the datasink report
- `local_output_dir` (*string*) – Local output / derivatives directory
- `debug` (*bool*) – If `True`, print output dictionary

Returns `dict_outputs` – Dictionary of pipeline outputs

Return type *dict*

`cmtklib.util.isavailable(file)`

Check if file is available and return the file if it is.

Used for debugging.

Parameters `file` (*File*) – Input file

Returns `file` – Output file

Return type *File*

`cmtklib.util.length(xyz, along=False)`

Euclidean length of track line.

Parameters

- `xyz` (*array-like shape (N, 3)*) – array representing x,y,z of N points in a track
- `along` (*bool, optional*) – If `True`, return array giving cumulative length along track, otherwise (default) return scalar giving total length.

Returns `L` – scalar in case of `along == False`, giving total length, array if `along == True`, giving cumulative lengths.

Return type scalar or array shape (N-1,)

Examples

```
>>> xyz = np.array([[1,1,1],[2,3,4],[0,0,0]])
>>> expected_lens = np.sqrt([1+2**2+3**2, 2**2+3**2+4**2])
>>> length(xyz) == expected_lens.sum()
True
>>> len_along = length(xyz, along=True)
>>> np.allclose(len_along, expected_lens.cumsum())
True
>>> length([])
0
>>> length([[1, 2, 3]])
0
>>> length([], along=True)
array([0])
```

`cmtklib.util.load_graphs(output_dir, subjects, parcellation_scheme, weight)`

Return a dictionary of connectivity matrices (graph adjacency matrices).

Still in development

Parameters

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

Returns `connmats` – Dictionary of connectivity matrices

Return type `dict`

`cmtklib.util.magn(xyz, n=1)`

Returns the vector magnitude

Parameters

- **xyz** (*vector*) – Input vector
- **n** (*int*) – Tile by n if n>1 before return

`cmtklib.util.mean_curvature(xyz)`

Calculates the mean curvature of a curve.

Parameters **xyz** (*array-like shape (N,3)*) – array representing x,y,z of N points in a curve

Returns **m** – float representing the mean curvature

Return type `float`

Examples

Create a straight line and a semi-circle and print their mean curvatures

```
>>> from dipy.tracking import metrics as tm
>>> import numpy as np
>>> x=np.linspace(0,1,100)
>>> y=0*x
>>> z=0*x
>>> xyz=np.vstack((x,y,z)).T
>>> m=tm.mean_curvature(xyz) #mean curvature straight line
>>> theta=np.pi*np.linspace(0,1,100)
>>> x=np.cos(theta)
>>> y=np.sin(theta)
>>> z=0*x
>>> xyz=np.vstack((x,y,z)).T
>>> m=tm.mean_curvature(xyz) #mean curvature for semi-circle
```

`cmtklib.util.print_blue(message)`

Print blue-colored message

Parameters `message` (*string*) – The string of the message to be printed

`cmtklib.util.print_error(message)`

Print red-colored error message

Parameters `message` (*string*) – The string of the message to be printed

`cmtklib.util.print_warning(message)`

Print yellow-colored warning message

Parameters `message` (*string*) – The string of the message to be printed

`cmtklib.util.return_button_style_sheet(image, image_disabled=None, verbose=False)`

Return Qt style sheet for QPushButton with image

Parameters

- **image** (*string*) – Path to image to use as icon when button is enabled
- **image_disabled** (*string*) – Path to image to use as icon when button is disabled
- **verbose** (*Bool*) – Print the style sheet if True Default: False

Returns `button_style_sheet` – Qt style sheet for QPushButton with image

Return type `string`

`cmtklib.util.unicode2str(text)`

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

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

Returns `out_str` – Output string

Return type `str`

5.7 Adopting Datalad for collaboration

Datalad is a powerful tool for the versioning and sharing of raw and processed data as well as for the tracking of data provenance (i.e. the recording on how data was processed). This page was created with the intention to share with the user how we adopted datalad to manage and process datasets with Connectome Mapper 3 in our lab, following the YODA principles to our best.

You may ask “What are the YODA principles?”. They are basic principles behind creating, sharing, and publishing reproducible, understandable, and open data analysis projects with DataLad.

For more details and tutorials on Datalad and YODA, please check the recent [Datalad Handbook](#) and the [YODA principles](#).

Happy Collaborative and Reproducible Connectome Mapping!

Note: This was tested on Ubuntu 16.04 with Datalad 0.14.0, its extensions datalad-container 1.1.2, datalad-neuroimaging 0.3.1, and git-annex 8.20210127.

5.7.1 Prerequisites

- Python3 must be installed with Datalad and all dependencies. You can use the conda environment `py37cmp-gui` for instance. See [Installation of py37cmp-gui](#) for more installation details.
- A recent version of `git-annex` and `liblzma` (included in `py37cmp-gui` for Ubuntu/Debian).
- Docker must be installed on systems running Connectome Mapper 3. See [Prerequisites of Connectome Mapper 3](#) for more installation instructions.

5.7.2 Copy BIDS dataset to server

Copy the raw BIDS dataset using `rsync`:

```
rsync -P -avz -e 'ssh' \  
--exclude 'derivatives' \  
--exclude 'code' \  
--exclude '.datalad' \  
--exclude '.git' \  
--exclude '.gitattributes' \  
/path/to/ds-example/* \  
<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-example
```

where:

- `-P` is used to show progress during transfer
- `-v` increases verbosity
- `-e` specifies the remote shell to use (ssh)
- `-a` indicates archive mode
- `-z` enables file data compression during the transfer
- `--exclude DIR_NAME` exclude the specified `DIR_NAME` from the copy

5.7.3 Remote datalad dataset creation on Server

Connect to Server

To connect with SSH:

```
ssh <SERVER_USERNAME>@<SERVER_IP_ADDRESS>
```

Creation of Datalad dataset

Go to the source dataset directory:

```
cd /archive/data/ds-example
```

Initialize the Datalad dataset:

```
datalad create -f -c text2git -D "Original example dataset on lab server" -d .
```

where:

- `-f` forces to create the datalad dataset if not empty
- `-c text2git` configures Datalad to use git to manage text file
- `-D` gives a brief description of the dataset
- `-d` specify the location where the Datalad dataset is created

Track all files contained in the dataset with Datalad:

```
datalad save -m "Source (Origin) BIDS dataset" --version-tag origin
```

where:

- `-m MESSAGE` is the description of the state or the changes made to the dataset
- `--version-tag` tags the state of the Dataset

Report on the state of dataset content:

```
datalad status -r
git log
```

5.7.4 Processing using the Connectome Mapper BIDS App on Alice's workstation

Processed dataset creation

Initialize a datalad dataset with the YODA procedure:

```
datalad create -c text2git -c yoda \
-D "Processed example dataset by Alice with CMP3" \
/home/alice/data/ds-example-processed
```

This will create a datalad dataset with:

- a code directory in your dataset

- three files for human consumption (README.md, CHANGELOG.md)
- everything in the code/ directory configured to be tracked by Git, not git-annex
- README.md and CHANGELOG.md configured in the root of the dataset to be tracked by Git
- Text files configured to be tracked by Git

Go to the created dataset directory:

```
cd /home/alice/data/ds-example-processed
```

Create the derivatives output directory:

```
mkdir derivatives
```

Raw BIDS dataset installation

Install the remove datalad dataset ds-example in /home/alice/data/ds-example-processed/input/:

```
datalad install -d . -s ssh://<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-  
example \  
/home/alice/data/ds-example-processed/input/
```

where:

- -s SOURCE specifies the URL or local path of the installation source

Get T1w and Diffusion images to be processed

For reproducibility, create and write datalad get commands to get_required_files_for_analysis.sh:

```
echo "datalad get input/sub-*/ses-*/anat/sub-*_T1w.nii.gz" > code/get_required_files_for_  
analysis.sh  
echo "datalad get input/sub-*/ses-*/dwi/sub-*_dwi.nii.gz" >> code/get_required_files_for_  
analysis.sh  
echo "datalad get input/sub-*/ses-*/dwi/sub-*_dwi.bvec" >> code/get_required_files_for_  
analysis.sh  
echo "datalad get input/sub-*/ses-*/dwi/sub-*_dwi.bval" >> code/get_required_files_for_  
analysis.sh
```

Save the script to the dataset's history:

```
datalad save -m "Add script to get the files required for analysis by Alice"
```

Execute the script:

```
sh code/get_required_files_for_analysis.sh
```


Link the container image with the dataset

Add Connectome Mapper's container image to the datalad dataset:

```
datalad containers-add connectomemapper-bidsapp-<VERSION_TAG> \
--url dhub://sebastientourbier/connectomemapper-bidsapp:<VERSION_TAG> \
-d . \
--call-fmt \
"docker run --rm -t \
  -v "$(pwd)/input":/bids_dir \
  -v "$(pwd)/code":/bids_dir/code \
  -v "$(pwd)/derivatives:/output_dir \
  -u "$(id -u)": "$(id -g)" \
  sebastientourbier/connectomemapper-bidsapp:<VERSION_TAG> {cmd}"
```

Note: `--call-fmt` specifies a custom docker run command. The current directory is assumed to be the BIDS root directory and retrieve with `"$(pwd)/input` and the output directory is inside the `derivatives/` folder.

Important: The name of the container-name registered to Datalad cannot have dot as character so that a `<VERSION_TAG>` of `v3.X.Y` would need to be rewritten as `v3-X-Y`

Copy existing reference pipeline configuration files to `code` folder:

```
cp /path/to/existing/ref_anatomical_config.json \
code/ref_anatomical_config.json
cp /path/to/existing/ref_diffusion_config.json \
code/ref_diffusion_config.json
```

Copy FreeSurfer license file to `code` folder:

```
cp /path/to/freesurfer/license.txt \
code/license.txt
```

Save the state of the dataset prior to analysis:

```
datalad save -m "Alice's test dataset on local \
workstation ready for analysis with connectomemapper-bidsapp:<VERSION_TAG>" \
--version-tag ready4analysis-<date>-<time>
```

Run Connectome Mapper with Datalad

Run Connectome Mapper on all subjects:

```
datalad containers-run --container-name connectomemapper-bidsapp-<VERSION_TAG> \
--input code/ref_anatomical_config.json \
--input code/ref_diffusion_config.json \
--output derivatives \
/bids_dir /output_dir participant \
--anat_pipeline_config '/bids_dir/{inputs[0]}' \
--dwi_pipeline_config '/bids_dir/{inputs[1]}'
```

Note: `datalad containers-run` will take of replacing the `{inputs[i]}` by the value specified by the `i --input` flag (Indexing start at 0).

Save the state:

```
datalad save -m "Alice's test dataset on local \
workstation processed by connectomemapper-bidsapp:<VERSION_TAG>, {Date/Time}" \
--version-tag processed-<date>-<time>
```

Report on the state of dataset content:

```
datalad status -r
git log
```

Configure a datalad dataset target on the Server

Create a remote dataset repository and configures it as a dataset sibling to be used as a publication target:

```
datalad create-sibling --name remote -d . \
<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-example-processed
```

See the documentation of `datalad create-sibling` command for more details.

Update the remote datalad dataset

Push the datalad dataset with data derivatives to the server:

```
datalad push -d . --to remote
```

Note: `--to remote` specifies the remote dataset sibling i.e. `ssh://<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-example-processed` previously configured.

Uninstall all files accessible from the remote

With DataLad we don't have to keep those inputs around so you can safely uninstall them without losing the ability to reproduce an analysis:

```
datalad uninstall input/sub-*/**
```

5.7.5 Local collaboration with Bob for Electrical Source Imaging

Processed dataset installation on Bob's workstation

Install the processed datalad dataset ds-example-processed in /home/bob/data/ds-example-processed:

```
datalad install -s ssh://<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-example-processed \
/home/bob/data/ds-example-processed
```

Go to datalad dataset clone directory:

```
cd /home/bob/data/ds-example-processed
```

Get connectome mapper output files (Brain Segmentation and Multi-scale Parcellation) used by Bob in his analysis

For reproducibility, write datalad get commands to get_required_files_for_analysis_by_bob.sh:

```
echo "datalad get derivatives/cmp/sub-*/ses-*/anat/sub-*_mask.nii.gz" \
> code/get_required_files_for_analysis_by_bob.sh
echo "datalad get derivatives/cmp/sub-*/ses-*/anat/sub-*_class-*_dseg.nii.gz" \
>> code/get_required_files_for_analysis_by_bob.sh
echo "datalad get derivatives/cmp/sub-*/ses-*/anat/sub-*_scale*_atlas.nii.gz" \
>> code/get_required_files_for_analysis_by_bob.sh
```

Save the script to the dataset's history:

```
datalad save -m "Add script to get the files required for analysis by Bob"
```

Execute the script:

```
sh code/get_required_files_for_analysis_by_bob.sh
```

Update derivatives

Update derivatives with data produced by Cartool:

```
cd /home/bob/data/ds-example
mkdir derivatives/cartool
cp [...]
```

Save the state:

```
datalad save -m "Bob's test dataset on local \
workstation processed by cartool:<CARTOOL_VERSION>, {Date/Time}" \
--version-tag processed-<date>-<time>
```

Report on the state of dataset content:

```
datalad status -r
git log
```

Update the remote datalad dataset

Update the remote datalad dataset with data derivatives:

```
datalad push -d . --to origin
```

Note: `--to origin` specifies the origin dataset sibling i.e. `ssh://<SERVER_USERNAME>@<SERVER_IP_ADDRESS>:/archive/data/ds-example-processed` from which it was cloned.

Uninstall all files accessible from the remote

Again, with DataLad we don't have to keep those inputs around so you can safely uninstall them without losing the ability to reproduce an analysis:

```
datalad uninstall derivatives/cmp/*  
datalad uninstall derivatives/freesurfer/*  
datalad uninstall derivatives/nipype/*
```

Authors Sebastien Tourbier

Version Revision: 2 (Last modification: 2021 Feb 23)

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:latest
```

If you prefer, you can still build the Singularity image on your side using one of the 2 methods described in [Conversion to a Singularity image](#).

A list of useful singularity command can be found in [Useful singularity commands](#). For more documentation about Singularity, please check the [official documentation website](#).

Happy Large-Scale Connectome Mapping!

5.8.1 Prerequisites

- Singularity must be installed. Check the [official documentation webpage](#) for installation instructions.

5.8.2 Running the singularity image

The following example shows how to call from the terminal the Singularity image of the CMP3 BIDS App to perform both anatomical and diffusion pipelines for sub-01, sub-02 and sub-03 of a BIDS dataset whose root directory is located at `${localDir}`:

```
$ singularity run --containall \
  --bind ${localDir}:/bids_dir --bind ${localDir}/derivatives:/output_dir \
  library://connectomicslab/default/connectomemapper-bidsapp:latest \
  /bids_dir /output_dir participant --participant_label 01 02 03 \
  --anat_pipeline_config /bids_dir/code/ref_anatomical_config.json \
  --dwi_pipeline_config /bids_dir/code/ref_diffusion_config.json \
  --fs_license /bids_dir/code/license.txt \
  --number_of_participants_processed_in_parallel 3
```

Note: As you can see, the `singularity run` command is slightly different from the `docker run`. The `docker` option flag `-v` is replaced by the `singularity --bind` to map local folders inside the container. Last but not least, while `docker` containers are executed in total isolation, `singularity` images **MUST** run with the option flag `--containall`. Otherwise your `$HOME` and `$TMP` directories or your local environment variables might be shared inside the container.

5.8.3 Conversion to a Singularity image

It actually exists two options for Docker to Singularity container image conversion. Let's say we want to store Singularity-compatible image file in `~/Softwares/singularity/`.

Option 1 (recommended): Using the Docker image `docker2singularity`

1. Build locally in a `/tmp/test` folder:

```
$ mkdir -p /tmp/test
$ docker run -v /var/run/docker.sock:/var/run/docker.sock \
  -v /tmp/test:/output --privileged -t --rm \
  singularityware/docker2singularity \
  --name cmp-v3.0.0.simg \
  sebastientourbier/connectomemapper-bidsapp:v3.0.0
```

2. Move the converted image `cmp-|release|` to the `~/Softwares/singularity` folder on the cluster (via `ssh` using `scp` for instance)

```
$ scp -v /tmp/test/cmp-v3.0.0.simg <user>@<cluster_url>:~/Softwares/singularity/
↪ cmp-v3.0.0.simg
```

Advantage(s): Has never failed

Disadvantage(s): Have to make a-priori the conversion locally on a workstation where `docker` is installed and then upload the converted image to the cluster

Option 2 : Using singularity directly

```
$ singularity build ~/Softwares/singularity/cmp-v3.0.0.simg \
  docker://sebastientourbier/connectomemapper-bidsapp:v3.0.0
```

This command will directly download the latest version release of the Docker image from the DockerHub and convert it to a Singularity image.

Advantage(s): Can be executed on the cluster directly

Disadvantage(s): Has shown to fail because of some docker / singularity version incompatibilities

5.8.4 Useful singularity commands

- Display a container's metadata:

```
$ singularity inspect ~/Softwares/singularity/cmp-v3.0.0.simg
```

- Clean cache:

```
$ singularity cache clean
```

Authors Sebastien Tourbier

Version Revision: 2 (Last modification: 2021 Jan 04)

5.9 BSD 3-Clause License

Copyright (C) 2009-2021, 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

Date: Dec XX, 2021

This version corresponds to the first official release of Connectome Mapper 3 (CMP3). It incorporates [Pull Request #88](#) (>450 commits) which includes the following changes.

New features

- CMP3 can take custom segmentation (brain, white-matter, gray-matter and CSF masks, Freesurfer's aparcaseg - used for ACT for PFT) and parcellation files as long as they comply to [BIDS Derivatives specifications](#), by providing the label value for the different entity in the filename. This has led to the creation of the new module `cmtklib.bids.io`, which provides different classes to represent the diversity of custom input BIDS-formatted files. ([PR #88](#))
- CMP3 generates generic label-index mapping tsv files along with the parcellation files, in accordance to [BIDS derivatives](#). This has led to the creation of the `CreateBIDSStandardParcellationLabelIndexMappingFile` and `CreateCMPParcellationNodeDescriptionFilesFromBIDSFile` interfaces, which allows us to create the BIDS label-index mapping file from the parcellation node description files employed by CMP3 (that includes `_FreeSurferColorLUT.txt` and `_dseg.graphml`), and vice versa.
- CMP3 provide python wrappers to the Docker and Singularity container images (`connectomemapper3_docker` and `connectomemapper3_singularity`) that will generate and execute the appropriate command to run the BIDS App. ([PR #109](#), [PR #115](#), [PR #130](#))

Major changes

- Lausanne2018 parcellation has completely replaced the old Lausanne2008 parcellation. In brief, the new parcellation was introduced to provide (1) symmetry of labels between hemispheres, and (2) a more optimal generation of the volumetric parcellation images, that now are generated at once from `annot` files. This fixes the issue of overwritten labels encountered by in the process of creating the Lausanne2008 parcellation. Any code and data related to Lausanne2008 has been removed. If one still wish to use this old parcellation scheme, one should use CMP3 (v3.0.0-RC4).

Output updates

- Directories for the derivatives produced by `cmp` (`cmp`, `freesurfer`, `nipype`) were renamed to `cmp-`, `freesurfer-`, and `nipype-` to comply with BIDS 1.4.0+. ([PR #3 \(fork\)](#))

Code refactoring

- Creation in `AnatomicalPipeline`, `DiffusionPipeline`, `fMRIPipeline` of `create_datagrabber_node()` and `create_datasinker_node()` methods to reduce the code in `create_workflow()`.

- The `run(command)` function of `cmp.bidsappmanager.core` has been moved to `cmtklib.process`, which is used by the python wrappers in `cmp.cli`.

Pipeline Improvements

- Better handle of existing Freesurfer outputs. In this case, CMP3 does not re-create the `mri/orig/001.mgz` and connect the reconall interface anymore.
- Creation of 5TT, gray-matter / white-matter interface, and partial volume maps images are performed in the preprocessing stage of the diffusion pipeline only if necessary

Code Style

- Clean code and remove a number of commented lines that are now obsolete. Code related to the connection of nodes in the Nipype Workflow adopts a specific format and are protected from being reformatted by BLACK with the `# fmt: off` and `# fmt: on` tags.

Documentation

- Add instructions to use custom segmentation and parcellation files as inputs.
- Add description in contributing page of format for code related to the connection of the nodes in a Nipype Workflow.
- Add instructions to use the python wrappers for running the BIDS App. (PR #115)
- Add notification about the removal of the old Lausanne2008 parcellation, and remove any other mentions in the documentation.

Software container

- Define multiple build stages in Dockerfile, which can be run in parallel at build with BUILDKIT. (PR #88)

Software development life cycle

- Update the list of outputs of circleci tests with the new names of directories produced by cmp in `output_dir/`.
- Following major changes in the pricing plans of CircleCI but also to improve its readability, `circleci/config.yml` has been dramatically refactored, including:
 - * Use BUILDKIT in docker build to take advantage of the multi-stage build
 - * Reordering and modularization of the tests:
 - tests 01-02 (Docker): anatomical pipeline for each parcellation scheme
 - tests 03-06 (Docker): diffusion pipeline for dipy/mrtrix deterministic/probabilistic tractography
 - tests 07-08 (Docker): fMRI pipeline for FLIRT and BBRegistration registrations
 - test 09 (Singularity): anatomical pipeline for Lausanne2018 scheme
 - Creation of commands for steps that are shared between jobs to reduce code duplication

(PR #88)

Contributors

- [Sebastien Tourbier](#)
- [Anil Tuncel](#)
- [Jakub Jancovic](#)
- [Jonathan Wirsich](#)

Please check the [main pull request 88](#) page for more details.

5.10.2 Version 3.0.0-RC4

Date: March 07, 2021

This version corresponds to the fourth and final release candidate of Connectome Mapper 3 (CMP3). It incorporates the relatively large [Pull Request #74](#) (~270 commits) which includes the following changes such that it marks the end of the release candidate phase.

New features

- CMP3 pipeline configuration files adopt JSON as new format. ([PR #76](#))
- CMP3 is compatible with PyPI for installation. ([PR #78](#))
- BIDS convention naming of data derived from parcellation atlas adopt now the new BIDS entity `atlas-<atlas_label>` to distinguish data derived from different parcellation atlases. The use of the entity `desc-<scale_label>` to distinguish between parcellation scale has been replaced by the use of the entity `res-<scale_label>`. ([PR #79](#))

Updates

- Content of `dataset_description.json` for each derivatives folder has been updated to conform to BIDS version 1.4.0. ([PR #79](#))

Code refactoring

- Major refactoring of the `cmtklib.config` module with the addition and replacement of a number of new methods to handle JSON configuration files. (See [full diff on GitHub](#)) Configuration files in the old INI format can be converted automatically with the help of the two new methods `check_configuration_format()` and `convert_config_ini_2_json()` to detect if configuration files are in the INI format and to make the conversion. ([PR #76](#))
- Major changes to make `cmp` and `cmpbidsappmanager` compatible with the Python Package Index (pip) for package distribution and installation. This includes the merge of `setup.py` and `setup_gui.py`, which have been merged into one `setup.py` and a major refactoring to make pip happy, as well as the creation of a new `cmp.cli` module, migration to `cmp.cli` module and refactoring of the scripts `connectomemapper3`, `showmatrix_gpickle`, and `cmpbidsappmanager` with correction of code style issues and addition of missing docstrings. ([PR #78](#))

Improvements

- Clean parameters to be saved in configuration files with the new API. ([PR #74](#))
- Clean output printed by the `cmpbidsappmanager` Graphical User Interface. ([PR #74](#))
- Add in `cmtklib.config` the three new functions `print_error`, `print_blue`, and `print_warning` to use different colors to differentiate general info (default color), error (red), command or action (blue), and highlight or warning (yellow). ([PR #74](#))
- Clean code and remove a number of commented lines that are now obsolete. ([PR #74](#), [PR #79](#))

Documentation

- Review usage and add a note regarding the adoption of the new JSON format for configuration files. ([PR #76](#))
- Update tutorial on using CMP3 and Datalad for collaboration. ([PR #77](#))
- Update installation instruction of `cmpbidsappmanager` using `pip install ..` ([PR #78](#))
- Update list of outputs following the new BIDS derivatives naming convention introduced. ([PR #79](#))

Bug fixes

- Correct attributes related to the diffusion imaging model type `multishell`. ([PR #74](#))

- Review code in `cmtklib/connectome.py` for saving functional connectome files in GRAPHML format. (PR #74)

Software Updates

- Update version of datalad and dependencies (PR #77):
 - `datalad[full]==0.13.0` to `datalad[full]==0.14.0`.
 - `datalad-container==0.3.1` to `datalad-container==1.1.2`.
 - `datalad_neuroimaging==0.2.0` to `datalad-neuroimaging==0.3.1`.
 - `git-annex=8.20200617` to `git-annex=8.20210127`.
 - `datalad-revolution` was removed.

Software development life cycle

- Improve code coverage by calling the methods `check_stages_execution()` and `fill_stages_outputs()` on each pipeline when executed with coverage. (PR #75)
- Improve code coverage by saving in test-01 structural connectome files in MAT and GRAPHML format. (PR #74)
- Improve code coverage by saving in test-07 functional connectome files in GRAPHML format. (PR #74)
- Update the list of outputs for all tests. (PR #74)
- Add `test-python-install` job that test the build and installation of `cmp` and `cmpbidsappmanager` packages compatible with `pip`. (PR #78)

Please check the [main pull request 74](#) page for more details.

5.10.3 Version 3.0.0-RC3

Date: February 05, 2021

This version corresponds to the third release candidate of Connectome Mapper 3. In particular, it integrates [Pull Request #62](#) which includes:

Updates

- MRtrix3 has been updated from `3.0_RC3_latest` to `3.0.2`.
- Numpy has been updated from `1.18.5` to `1.19.2`.
- Nipype has been updated to `1.5.0` to `1.5.1`.
- Dipy has been updated from `1.0.0` to `1.3.0`.
- CVXPY has been updated from `1.1.5` to `1.1.7`.

Documentation

- Update outdated screenshots for GUI documentation page at [readthedocs](#) reported at [CMTK user-group](#).
- Correction of multiple typos.

Bug fixes

- Update code for Dipy tracking with DTI model following major changes in Dipy 1.0 (Fix reported issue #54).
- Update to Dipy 1.3.0 has removed the deprecated warnings related to CVXPY when using `MAP_MRI` (#63)
- Do not set anymore `OMP_NUM_THREADS` at execution due to allocation errors raised when using numpy function `dot` in Dipy.

Software development life cycle

- Add Test 08 that runs anatomical and fMRI pipelines with: Lausanne2018 parcellation, FSL FLIRT co-registration, all nuisance regression, linear detrending and scrubbing
- Add Test 09 that runs anatomical and dMRI pipelines with: Lausanne2018 parcellation, FSL FLIRT, Dipy SHORE, MRtrix SD_Stream tracking, MRtrix SIFT tractogram filtering
- Remove `deploy_singularity_latest` from the workflow for the sake of space on Sylabs.io.

Please check the [main pull request 62](#) page for more details.

5.10.4 Version 3.0.0-RC2-patch1

Date: February 4, 2021

This version fixes bugs in the second release candidate of Connectome Mapper 3 (v3.0.0-RC2). In particular, it includes:

Bug fixes

- Fix the error to save connectome in GraphML format reported in [#65](#) and ([Pull Request #66](#)).

Software development life cycle

- Remove publication of the Singularity image to sylabs.io when the master branch is updated for the sake of space (11GB limit)

Commits

- CI: remove publication of latest tag image on sylabs.io for space (2 days ago) - commit c765f79
- Merge pull request #66 from connectomicslab/v3.0.0-RC2-hotfix1 (3 days ago) - commit 0a2603e
- FIX: update g2.node to g2.nodes when saving connectomes as graphml (fix #65) (6 days ago) - commit d629eef
- FIX: enabled/disabled gray-out button “Run BIDS App” with Qt Style sheet [skip ci] (3 weeks ago) - commit 10e78d9
- MAINT: removed commented lines in cmpbidsappmanager/gui.py [skip ci] (3 weeks ago) - commit 4cc11e7
- FIX: check availability of modalities in the BIDS App manager window [skip ci] (3 weeks ago) - commit 80fbee2
- MAINT: update copyright year [skip ci] (3 weeks ago) - commit f7d0ffb
- CI: delete previous container with latest TAG on sylabs.io [skip ci] (4 weeks ago) - commit 15c9b18
- DOC: update tag to latest in runonhpc.rst [skip ci] (4 weeks ago) - commit 3165bcc
- CI: comment lines related to version for singularity push (4 weeks ago) - commit 3952d46

5.10.5 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.

5.10.6 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 sty 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.

5.10.7 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.

5.10.8 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.

5.10.9 Version 3.0.0-beta-20200227

Date: February 27, 2020

This version addresses multiple issues to make successful conversion and run of the CMP3 BIDS App on HPC (Clusters) using Singularity.

- Revised the build of the master and BIDS App images:
 - Install locales and set `$LC_ALL` and `$LANG` to make freesurfer hippocampal subfields and brainstem segmentation (matlab-based) modules working when run in the converted SIngularity image
 - BIDS input and output directories inside the BIDS App container are no longer the `/tmp` and `/tmp/derivatives` folders but `/bids_dir` and `/output_dir`. .. warning:: this might affect the use of Datalad container (To be confirmed.)

- Fix the branch of mrtrix3 to check out
- Updated metadata
- Fix the configuration of CircleCI to not use Docker layer cache feature anymore as this feature is not included anymore in the free plan for open source projects.
- Improved documentation where the latest version should be dynamically generated everywhere it should appear.

5.10.10 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)

5.10.11 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 your 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 (Version v3.0.0). 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



Contact:
sebastien.tourbier@unil.ch

Connectome Mapper 3: a software pipeline for multi-scale connectome mapping of multimodal MR data

S. Tourbier¹, Y. Alemán-Gómez^{1,4}, E. Mullier¹, A. Griffa², M. Bach Cuadra^{1,3}, P. Hagmann¹

1. Department of Radiology, University Hospital of Lausanne (CHUV) and University of Lausanne (UNIL), Lausanne, Switzerland
2. Medical Image Processing Lab (MIPLAB), Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
3. Medical Image Analysis Laboratory (MIAL), Centre d'Imagerie BioMédicale (CIBM), Lausanne, Switzerland

OHBM 2020
A Virtual Experience for Engaging Minds & Empowering Brain Science

PNB
Brain Network Science

Supported in part by the Swiss National Science Foundation under Sinergia Grant under Grant CRSI1_170873.

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.

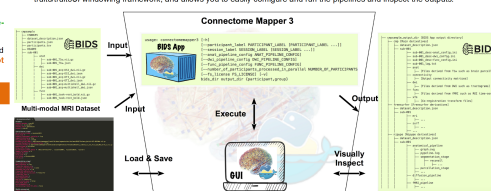
OBJECTIFS

- 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 [3] and processing [4,5] that:
 - promote processing interoperability, reusability, portability and reproducibility, and
 - have proven to be capable of effective large scale collaboration

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]) on HPC, 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 JupyterLab windowing framework, and allows you to easily configure and run the pipelines and inspect the outputs.



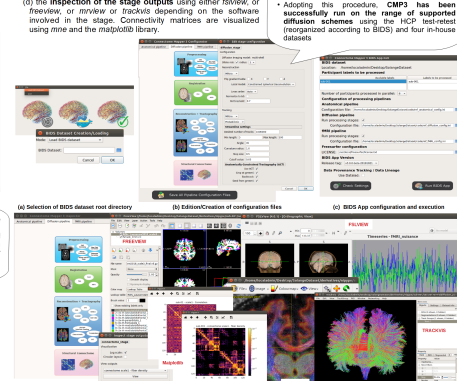
GRAPHICAL USER INTERFACE

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 **RAWVIEW**, or **Freeview**, or **minview** or **trackvis** depending on the software involved in the stage. Connectivity matrices are visualized using **min** 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.

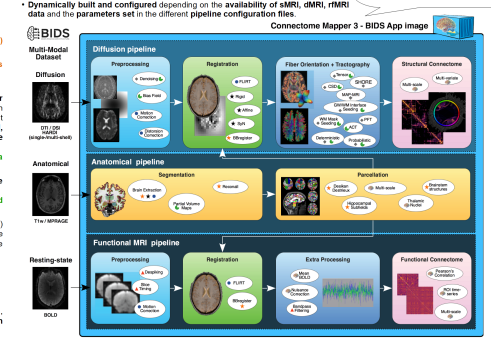


HIGHLIGHTS

- Extend the 5-scale brain gray matter parcellation [2] derived from the Desikan-Killiany atlas [6] with:
 - a subdivision of the thalamus into 7 nuclei [7],
 - the hippocampus into 12 subfields [8] and
 - the brainstem into 4 sub-structures [9].
- unique multi-scale hierarchical brain parcellation
- 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 **NiPytype** [4] to implement a modular processing workflow composed of configurable pipelines, each dedicated to one modality (anatomical, diffusion, fMRI), that interface with tools from **PSL**[10], **FreeSurfer**[11], **ANTs**[12], **Dipy**[13], **MRtrix3**[14], **AFNI**[15], and the **in-house Connectome Mapper library (cmLib)**
 - 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.
 - accessible, easy-to-be-installed, easy-to-use
- Distributed under the 3-Clause BSD license
- Hosted on **GitHub** @ <https://github.com/ReproNim/connectomemapper> where issues and new feature requests are transparently discussed, and versions are released through continuous integration testing
 - modern community-driven software practices

PROCESSING WORKFLOW

Thanks to its modular architecture and the use of **NiPytype** and **traits** libraries, new pipelines and stages with GUI components can be added with relatively little effort to account for additional imaging modalities and algorithms.



Connectome Mapper 3 - BIDS App image

Levers: **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 CircleCI. 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 **ReproNim/Sinergia 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 CMP3 User Group forum.

[1] Oudizien et al., PLoS ONE 2013; [2] Cammoun et al., J. Neurosci. Methods 2012; [3] Gorgolewski et al., Scientific Data 2016; [4] Gorgolewski et al., Front. Neurosci. 2011; [5] Gorgolewski et al., PLoS CB 2017; [6] Desikan et al., Neuroimage 2004; [7] Hagmann et al., Scientific Data 2018; [8] Iglesias et al., Neuroimage 2015; [9] Iglesias et al., Neuroimage 2016; [10] Jenkinson et al., ImageMag 2012; [11] Fennell et al., Neuroimage 2012; [12] Avants et al., MIA 2018; [13] Cammoun et al., Front. Neuroinform. 2014; [14] Tourner et al., Neuroimage 2019; [15] Cox et al., Computers and Biomedical Research 1996; [16] Mehner et al., Linux. Journal 2014; [17] Kaufman et al., PLoS ONE 2017.

5.12 Contributors

5.13 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*

5.13.1 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 CMP3 and are not listed as contributor, please add yourself and make a pull request.

This also means that further development, typically additions of other tools and configuration options should go in this direction.

5.13.2 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 :)

5.13.3 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 scripts/build_bidsapp.sh
```

Note: Tag of the version of the image is extracted from `cmp/info.py`. You might want to change the version in this file to not overwrite an other existing image with the same version.

How to build the documentation locally

To generate the documentation:

1. Install the CMP3 conda environment `py37cmp-gui`:

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

2. Activate CMP3 conda environment `py37cmp-gui`:

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

3. Install all dependencies such as sphinx and its extensions, required for the build:

```
(py37cmp-gui)$ pip install -r docs/requirements.txt
```

4. Install `connectomemapper3`:

```
(py37cmp-gui)$ pip install .
```

5. Run the script `scripts/build_docs.sh` to generate the HTML documentation in `docs/_build/html`:

```
(py37cmp-gui)$ sh scripts/build_docs.sh
```

Note: Make sure to have (1) activated the conda environment `py37cmp-gui` and (2) reinstalled `connectomemapper3` with `pip` before running `build_docs.sh`.

Authors Sebastien Tourbier, Adrien Birbaumer

Version Revision: 2

Acknowledgments

We thanks the authors of [these great contributing guidelines](#), from which part of this document has been inspired and adapted.

5.14 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. “Continuous diffusion signal, EAP and ODF estimation via Compressive Sensing in diffusion MRI”
- [Smith2013SIFT] R.E. Smith et al., NeuroImage 67 (2013), pp. 298–312, <<https://www.ncbi.nlm.nih.gov/pubmed/23238430>>.
- [Smith2015SIFT2] Smith RE et al., Neuroimage, 2015, 119:338-51. <<https://doi.org/10.1016/j.neuroimage.2015.06.092>>.
- [Iglesias2015Brainstem] Iglesias et al., NeuroImage, 113, June 2015, 184-195. <http://www.nmr.mgh.harvard.edu/~iglesias/pdf/Neuroimage_2015_brainstem.pdf>
- [Iglesias2015Hippo] Iglesias et al., Neuroimage, 115, July 2015, 117-137. <<http://www.nmr.mgh.harvard.edu/~iglesias/pdf/subfieldsNeuroimage2015preprint.pdf>>
- [Najdenovska18] Najdenovska et al., Sci Data 5, 180270 (2018). <<https://doi.org/10.1038/sdata.2018.270>>

PYTHON MODULE INDEX

C

cmp, 82
 cmp.bidsappmanager.pipelines.anatomical, 127
 cmp.bidsappmanager.pipelines.anatomical.anatomical, 127
 cmp.bidsappmanager.pipelines.diffusion, 128
 cmp.bidsappmanager.pipelines.diffusion.diffusion, 128
 cmp.bidsappmanager.pipelines.functional, 129
 cmp.bidsappmanager.pipelines.functional.eeg, 129
 cmp.bidsappmanager.pipelines.functional.fMRI, 130
 cmp.bidsappmanager.project, 122
 cmp.bidsappmanager.stages, 131
 cmp.bidsappmanager.stages.connectome, 131
 cmp.bidsappmanager.stages.connectome.connectome, 131
 cmp.bidsappmanager.stages.connectome.fmri_connectome, 132
 cmp.bidsappmanager.stages.diffusion, 133
 cmp.bidsappmanager.stages.diffusion.diffusion, 133
 cmp.bidsappmanager.stages.diffusion.reconstruction, 134
 cmp.bidsappmanager.stages.diffusion.tracking, 134
 cmp.bidsappmanager.stages.functional, 135
 cmp.bidsappmanager.stages.functional.functionalMRI, 135
 cmp.bidsappmanager.stages.parcellation, 136
 cmp.bidsappmanager.stages.parcellation.parcellation, 136
 cmp.bidsappmanager.stages.preprocessing, 137
 cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing, 137
 cmp.bidsappmanager.stages.preprocessing.preprocessing, 137
 cmp.bidsappmanager.stages.registration, 138
 cmp.bidsappmanager.stages.registration.registration, 138
 cmp.bidsappmanager.stages.segmentation, 139
 cmp.bidsappmanager.stages.segmentation.segmentation, 139
 cmp.parser, 82
 cmp.pipelines.anatomical, 89
 cmp.pipelines.anatomical.anatomical, 89
 cmp.pipelines.common, 88
 cmp.pipelines.diffusion, 91
 cmp.pipelines.diffusion.diffusion, 91
 cmp.pipelines.functional, 94
 cmp.pipelines.functional.eeg, 94
 cmp.pipelines.functional.fMRI, 95
 cmp.project, 83
 cmp.stages, 97
 cmp.stages.common, 121
 cmp.stages.connectome, 97
 cmp.stages.connectome.connectome, 97
 cmp.stages.connectome.fmri_connectome, 98
 cmp.stages.diffusion, 100
 cmp.stages.diffusion.diffusion, 100
 cmp.stages.diffusion.reconstruction, 103
 cmp.stages.diffusion.tracking, 105
 cmp.stages.functional, 108
 cmp.stages.functional.functionalMRI, 108
 cmp.stages.parcellation, 109
 cmp.stages.parcellation.parcellation, 109
 cmp.stages.preprocessing, 111
 cmp.stages.preprocessing.fmri_preprocessing, 111
 cmp.stages.preprocessing.preprocessing, 112
 cmp.stages.registration, 115
 cmp.stages.registration.registration, 115
 cmp.stages.segmentation, 119
 cmp.stages.segmentation.segmentation, 119
 cmcklib, 140
 cmcklib.bids, 140
 cmcklib.bids.io, 140
 cmcklib.bids.utils, 142
 cmcklib.config, 194
 cmcklib.connectome, 198
 cmcklib.diffusion, 201
 cmcklib.functionalMRI, 207
 cmcklib.interfaces, 143

- `cmtklib.interfaces.afni`, [143](#)
- `cmtklib.interfaces.ants`, [146](#)
- `cmtklib.interfaces.camino`, [147](#)
- `cmtklib.interfaces.camino2trackvis`, [150](#)
- `cmtklib.interfaces.diffusion_toolkit`, [151](#)
- `cmtklib.interfaces.dipy`, [155](#)
- `cmtklib.interfaces.freesurfer`, [161](#)
- `cmtklib.interfaces.fsl`, [164](#)
- `cmtklib.interfaces.misc`, [173](#)
- `cmtklib.interfaces.mrtrix3`, [174](#)
- `cmtklib.parcellation`, [211](#)
- `cmtklib.util`, [218](#)

A

`acquisition` (`cmtklib.bids.io.CustomBIDSFile` attribute), 141
`act_tracking` (`cmp.stages.preprocessing.preprocessing.PreprocessingConfig` attribute), 114
`act_tracking` (`cmp.stages.registration.registration.RegistrationConfig` attribute), 117
`aggregate_outputs()` (`cmtklib.interfaces.fsl.Orient` method), 173
`anat_available_config` (`cmp.project.CMP_Project_Info` attribute), 85
`anat_config_error_msg` (`cmp.project.CMP_Project_Info` attribute), 84
`anat_config_to_load` (`cmp.project.CMP_Project_Info` attribute), 85
`anat_custom_last_stage` (`cmp.project.CMP_Project_Info` attribute), 85
`anat_flow` (`cmp.pipelines.common.Pipeline` attribute), 88
`anat_inputs_checked` (`cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler` attribute), 123
`anat_inputs_checked` (`cmp.bidsappmanager.project.CMP_MainWindowHandler` attribute), 125
`anat_load_config_json()` (in module `cmtklib.config`), 194
`anat_outputs_checked` (`cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler` attribute), 123
`anat_outputs_checked` (`cmp.bidsappmanager.project.CMP_MainWindowHandler` attribute), 125
`anat_pipeline` (`cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler` attribute), 123
`anat_pipeline` (`cmp.bidsappmanager.project.CMP_MainWindowHandler` attribute), 125
`anat_run` (`cmp.project.CMP_Project_Info` attribute), 84
`anat_runs` (`cmp.project.CMP_Project_Info` attribute), 83
`anat_save_config()` (in module `cmtklib.config`), 194
`anat_stage_names` (`cmp.project.CMP_Project_Info` attribute), 85
`anatomical_processed` (`cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler` attribute), 123
`anatomical_processed` (`cmp.bidsappmanager.project.CMP_MainWindowHandler` attribute), 125
`AnatomicalPipeline` (class in `cmp.pipelines.anatomical.anatomical`), 89
`AnatomicalPipelineUI` (class in `cmp.bidsappmanager.pipelines.anatomical.anatomical`), 127
`angle` (`cmp.stages.diffusion.tracking.MRtrix_tracking_config` attribute), 107
`ants_bspline_interpolation_parameters` (`cmp.stages.registration.registration.RegistrationConfig` attribute), 115
`ants_convergence_thresh` (`cmp.stages.registration.registration.RegistrationConfig` attribute), 116
`ants_convergence_winsize` (`cmp.stages.registration.registration.RegistrationConfig` attribute), 116
`ants_gauss_interpolation_parameters` (`cmp.stages.registration.registration.RegistrationConfig` attribute), 115
`ants_interpolation` (`cmp.stages.registration.registration.RegistrationConfig` attribute), 115
`ants_linear_cost` (`cmp.stages.registration.registration.RegistrationConfig` attribute), 116
`ants_linear_gradient_step` (`cmp.stages.registration.registration.RegistrationConfig` attribute), 116
`ants_linear_sampling_perc` (`cmp.stages.registration.registration.RegistrationConfig` attribute), 116
`ants_linear_sampling_strategy` (`cmp.stages.registration.registration.RegistrationConfig` attribute), 116

- attribute), 116
- ants_lower_quantile (cmp.stages.registration.registration.RegistrationConfig attribute), 115
- ants_multilab_interpolation_parameters (cmp.stages.registration.registration.RegistrationConfig attribute), 115
- ants_nonlinear_cost (cmp.stages.registration.registration.RegistrationConfig attribute), 116
- ants_nonlinear_gradient_step (cmp.stages.registration.registration.RegistrationConfig attribute), 116
- ants_nonlinear_total_field_variance (cmp.stages.registration.registration.RegistrationConfig attribute), 116
- ants_nonlinear_update_field_variance (cmp.stages.registration.registration.RegistrationConfig attribute), 116
- ants_perform_syn (cmp.stages.registration.registration.RegistrationConfig attribute), 116
- ants_precision_type (cmp.stages.parcellation.parcellation.ParcellationConfig attribute), 110
- ants_probmaskfile (cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 119
- ants_regmaskfile (cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 119
- ants_templatefile (cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 119
- ants_upper_quantile (cmp.stages.registration.registration.RegistrationConfig attribute), 116
- apply_scrubbing (cmp.stages.connectome.fmri_connectome.ConnectomeConfig attribute), 98
- apply_to_eroded_brain (cmp.stages.registration.registration.RegistrationConfig attribute), 117
- apply_to_eroded_csf (cmp.stages.registration.registration.RegistrationConfig attribute), 117
- apply_to_eroded_wm (cmp.stages.registration.registration.RegistrationConfig attribute), 117
- ApplymultipleMRConvert, 174
- ApplymultipleMRCrop, 175
- ApplymultipleMRTransforms, 175
- ApplymultipleWarp, 164
- ApplymultipleXfm, 164
- atlas (cmtklib.bids.io.CustomBIDSFile attribute), 141
- atlas_info (cmp.project.CMP_Project_Info attribute), 84
- atlas_info (cmp.stages.parcellation.parcellation.ParcellationConfig attribute), 110
- ## B
- backtrack (cmp.stages.diffusion.tracking.MRtrix_tracking_config attribute), 107
- Bandpass, 143
- base_directory (cmp.project.CMP_Project_Info attribute), 83
- base_name (cmtklib.diffusion.Make_Mrtrix_Seeds attribute), 204
- base_name (cmtklib.diffusion.Make_Seeds attribute), 205
- BBRegister, 161
- BColors (class in cmtklib.util), 218
- bias_field_algo (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 113
- bias_field_correction (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 113
- bids_dir (cmp.stages.common.Stage attribute), 122
- bids_layout (cmp.project.CMP_Project_Info attribute), 83
- bids_session_label (cmp.stages.common.Stage attribute), 121
- bids_subject_label (cmp.stages.common.Stage attribute), 121
- big_delta (cmp.stages.diffusion.reconstruction.Dipy_recon_config attribute), 104
- BinaryThreshold, 165
- BOLD (cmtklib.util.BColors attribute), 218
- brain_mask_extraction_tool (cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 119
- ## C
- camino2ftrackvis, 150
- check_config() (cmp.pipelines.anatomical.anatomical.AnatomicalPipeline method), 89
- check_config() (cmp.pipelines.common.Pipeline method), 88
- check_config() (cmp.pipelines.diffusion.diffusion.DiffusionPipeline method), 91
- check_config() (cmp.pipelines.functional.fMRI.fMRIPipeline method), 95
- check_configuration_format() (in module cmtklib.config), 195
- check_configuration_version() (in module cmtklib.config), 195
- check_directory_exists() (in module cmtklib.util), 218
- check_input() (cmp.bidsappmanager.pipelines.anatomical.anatomical.AnatomicalPipeline method), 128
- check_input() (cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipeline method), 129
- check_input() (cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipeline method), 130

`check_input()` (*cmp.pipelines.anatomical.anatomical.AnatomicalPipeline* module), 89
`check_input()` (*cmp.pipelines.diffusion.diffusion.DiffusionPipeline* module), 91
`check_input()` (*cmp.pipelines.functional.fMRI.fMRIPipeline* module), 95
`Check_Input_Notification` (class in *cmp.pipelines.anatomical.anatomical*), 90
`check_output()` (*cmp.bidsappmanager.pipelines.anatomical.anatomical.AnatomicalPipelineUI* module), 128
`check_output()` (*cmp.pipelines.anatomical.anatomical.AnatomicalPipeline* module), 90
`check_settings()` (*cmp.bidsappmanager.project.CMP_BIDSAppManagerUIHandler* module), 123
`check_stages_execution()` (*cmp.pipelines.common.Pipeline* method), 88
`circular_layout` (*cmp.bidsappmanager.stages.connectome.connectome.ConnectomeStagesUI* attribute), 131
`circular_layout` (*cmp.bidsappmanager.stages.connectome.connectome.ConnectomeStagesUI* attribute), 132
`circular_layout` (*cmp.stages.connectome.connectome.ConnectomeStagesUI* attribute), 97
`circular_layout` (*cmp.stages.connectome.fmri_connectome.fmri_connectome.FMRIConnectomeStagesUI* attribute), 99
`clean_cache()` (in *cmp.bidsappmanager.project* module), 126
`clear_stages_outputs()` (*cmp.pipelines.common.Pipeline* method), 88
`cmat()` (in module *cmtklib.connectome*), 200
`cmp` module, 82
`cmp.bidsappmanager.pipelines.anatomical` module, 127
`cmp.bidsappmanager.pipelines.anatomical.anatomical` module, 127
`cmp.bidsappmanager.pipelines.diffusion` module, 128
`cmp.bidsappmanager.pipelines.diffusion.diffusion` module, 128
`cmp.bidsappmanager.pipelines.functional` module, 129
`cmp.bidsappmanager.pipelines.functional.eeg` module, 129
`cmp.bidsappmanager.pipelines.functional.fMRI` module, 130
`cmp.bidsappmanager.project` module, 122
`cmp.bidsappmanager.stages` module, 131
`cmp.bidsappmanager.stages.connectome` module, 131
`cmp.bidsappmanager.stages.connectome.connectome` module, 131
`cmp.bidsappmanager.stages.connectome.fmri_connectome` module, 132
`cmp.bidsappmanager.stages.diffusion` module, 133
`cmp.bidsappmanager.stages.diffusion.diffusion` module, 133
`cmp.bidsappmanager.stages.diffusion.reconstruction` module, 134
`cmp.bidsappmanager.stages.diffusion.tracking` module, 134
`cmp.bidsappmanager.stages.functional` module, 135
`cmp.bidsappmanager.stages.functional.functionalMRI` module, 135
`cmp.bidsappmanager.stages.parcellation` module, 136
`cmp.bidsappmanager.stages.parcellation.parcellation` module, 136
`cmp.bidsappmanager.stages.preprocessing` module, 137
`cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing` module, 137
`cmp.bidsappmanager.stages.preprocessing.preprocessing` module, 137
`cmp.bidsappmanager.stages.registration` module, 138
`cmp.bidsappmanager.stages.registration.registration` module, 138
`cmp.bidsappmanager.stages.segmentation` module, 139
`cmp.bidsappmanager.stages.segmentation.segmentation` module, 139
`cmp.parser` module, 82
`cmp.pipelines.anatomical` module, 89
`cmp.pipelines.anatomical.anatomical` module, 89
`cmp.pipelines.common` module, 88
`cmp.pipelines.diffusion` module, 91
`cmp.pipelines.diffusion.diffusion` module, 91
`cmp.pipelines.functional` module, 94
`cmp.pipelines.functional.eeg` module, 94
`cmp.pipelines.functional.fMRI` module, 95
`cmp.project` module, 83
`cmp.stages` module, 83

- module, 97
- cmp.stages.common
 - module, 121
- cmp.stages.connectome
 - module, 97
- cmp.stages.connectome.connectome
 - module, 97
- cmp.stages.connectome.fmri_connectome
 - module, 98
- cmp.stages.diffusion
 - module, 100
- cmp.stages.diffusion.diffusion
 - module, 100
- cmp.stages.diffusion.reconstruction
 - module, 103
- cmp.stages.diffusion.tracking
 - module, 105
- cmp.stages.functional
 - module, 108
- cmp.stages.functional.functionalMRI
 - module, 108
- cmp.stages.parcellation
 - module, 109
- cmp.stages.parcellation.parcellation
 - module, 109
- cmp.stages.preprocessing
 - module, 111
- cmp.stages.preprocessing.fmri_preprocessing
 - module, 111
- cmp.stages.preprocessing.preprocessing
 - module, 112
- cmp.stages.registration
 - module, 115
- cmp.stages.registration.registration
 - module, 115
- cmp.stages.segmentation
 - module, 119
- cmp.stages.segmentation.segmentation
 - module, 119
- CMP_BIDSAppWindowHandler (class *cmp.bidsappmanager.project*), 122
- CMP_ConfigQualityWindowHandler (class *cmp.bidsappmanager.project*), 123
- CMP_MainWindowHandler (class *cmp.bidsappmanager.project*), 125
- CMP_Project_Info (class in *cmp.project*), 83
- CMTK_cmat, 198
- CMTK_rsfmri_cmat, 199
- cmtklib
 - module, 140
- cmtklib.bids
 - module, 140
- cmtklib.bids.io
 - module, 140

- cmtklib.bids.utils
 - module, 142
- cmtklib.config
 - module, 194
- cmtklib.connectome
 - module, 198
- cmtklib.diffusion
 - module, 201
- cmtklib.functionalMRI
 - module, 207
- 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, 151
- cmtklib.interfaces.dipy
 - module, 155
- cmtklib.interfaces.freesurfer
 - module, 161
- cmtklib.interfaces.fsl
 - module, 164
- cmtklib.interfaces.misc
 - module, 173
- cmtklib.interfaces.mrtrix3
 - module, 174
- cmtklib.parcellation
 - module, 211
- cmtklib.util
 - module, 218
- CombineParcellations, 211
- compute_curvature (*cmp.stages.connectome.connectome.ConnectomeCo*
attribute), 97
- in compute_curvature_array() (in module *cmtk-*
lib.connectome), 201
- in compute_length_array() (in module *cmtk-*
lib.diffusion), 207
- in ComputeParcellationRoiVolumes, 212
- ConcatOutputsAsTuple, 173
- config (*cmp.stages.common.Stage attribute*), 122
- config_view (*cmp.bidsappmanager.stages.connectome.connectome.Conne*
attribute), 131
- config_view (*cmp.bidsappmanager.stages.connectome.fmri_connectome.C*
attribute), 132
- config_view (*cmp.bidsappmanager.stages.diffusion.diffusion.DiffusionSta*
attribute), 133
- config_view (*cmp.bidsappmanager.stages.functional.functionalMRI.Func*
attribute), 135

`config_view(cmp.bidsappmanager.stages.parcellation.parcellation.ParcellationStageUI attribute), 136`
`create_datagrabber_node()`
`config_view(cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing.DiffusionPipeline attribute), 137`
`method), 92`
`config_view(cmp.bidsappmanager.stages.preprocessing.parcellation.ParcellationStageUI attribute), 138`
`(cmp.pipelines.functional.fMRI.fMRIPipeline`
`config_view(cmp.bidsappmanager.stages.registration.registration.RegistrationStageUI attribute), 139`
`create_datasinker_node()`
`config_view(cmp.bidsappmanager.stages.segmentation.segmentation.SegmentationStageUI attribute), 140`
`method), 90`
`connectivity_metrics`
`create_datasinker_node()`
`(cmp.bidsappmanager.stages.connectome.connectome.ConnectomeConfigUI attribute), 131`
`method), 92`
`connectivity_metrics`
`create_datasinker_node()`
`(cmp.stages.connectome.connectome.ConnectomeConfig (cmp.pipelines.functional.fMRI.fMRIPipeline`
`attribute), 97`
`method), 96`
`connectome(cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipelineFlow() (in module`
`attribute), 129`
`cmp.stages.diffusion.reconstruction), 105`
`connectome(cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipelineTrackingFlow() (in module`
`attribute), 130`
`cmp.stages.diffusion.tracking), 107`
`ConnectomeConfig (class in create_endpoints_array() (in module cmtk-`
`cmp.stages.connectome.connectome), 97`
`lib.connectome), 201`
`ConnectomeConfig (class in create_field_template_dict()`
`cmp.stages.connectome.fmri_connectome), (cmp.pipelines.diffusion.diffusion.DiffusionPipeline`
`98`
`method), 92`
`ConnectomeConfigUI (class in create_mrtrix_recon_flow() (in module`
`cmp.bidsappmanager.stages.connectome.connectome), cmp.stages.diffusion.reconstruction), 105`
`131`
`create_mrtrix_tracking_flow() (in module`
`ConnectomeConfigUI (class in cmp.stages.diffusion.tracking), 107`
`cmp.bidsappmanager.stages.connectome.fmri_connectome), create_pipeline_flow()`
`132`
`(cmp.pipelines.anatomical.anatomical.AnatomicalPipeline`
`ConnectomeStage (class in method), 90`
`cmp.stages.connectome.connectome), 98`
`create_pipeline_flow()`
`ConnectomeStage (class in (cmp.pipelines.diffusion.diffusion.DiffusionPipeline`
`cmp.stages.connectome.fmri_connectome), method), 92`
`99`
`create_pipeline_flow()`
`ConnectomeStageUI (class in (cmp.pipelines.functional.fMRI.fMRIPipeline`
`cmp.bidsappmanager.stages.connectome.connectome), method), 96`
`131`
`create_roi() (in module cmtklib.parcellation), 216`
`ConnectomeStageUI (class in create_stage_flow()`
`cmp.bidsappmanager.stages.connectome.fmri_connectome), (cmp.pipelines.common.Pipeline method),`
`132`
`88`
`ConstrainedSphericalDeconvolution, 176`
`create_subject_configuration_from_ref() (in`
`contrast_type(cmp.stages.registration.registration.RegistrationConfig module cmtklib.config), 195`
`attribute), 117`
`create_T1_and_Brain() (in module cmtk-`
`convert_config_ini_2_json() (in module cmtk- lib.parcellation), 216`
`lib.config), 195`
`create_wm_mask() (in module cmtklib.parcellation),`
`convertList2Tuple() (in module cmtklib.util), 218`
`216`
`copyBrainMaskToFreesurfer, 163`
`create_workflow() (cmp.stages.connectome.connectome.ConnectomeStage`
`copyFileToFreesurfer, 163`
`method), 98`
`create_configparser_from_pipeline() (in module create_workflow() (cmp.stages.connectome.fmri_connectome.ConnectomeStage`
`cmtklib.config), 195`
`method), 99`
`create_datagrabber_node() create_workflow() (cmp.stages.diffusion.diffusion.DiffusionStage`
`(cmp.pipelines.anatomical.anatomical.AnatomicalPipeline method), 101, 102`

[create_workflow\(\) \(cmp.stages.functional.functionalMRI.FunctionalMRIStage attribute\), 108, 109](#)
[create_workflow\(\) \(cmp.stages.parcellation.parcellation.ParcellationStage attribute\), 110](#)
[create_workflow\(\) \(cmp.stages.preprocessing.fmri_preprocessing.PreprocessingStage attribute\), 112](#)
[create_workflow\(\) \(cmp.stages.preprocessing.preprocessing.PreprocessingStage attribute\), 114](#)
[create_workflow\(\) \(cmp.stages.registration.registration.RegistrationStage attribute\), 118](#)
[create_workflow\(\) \(cmp.stages.segmentation.segmentation.SegmentationStage attribute\), 120](#)
[create_workflow_custom\(\) \(cmp.stages.parcellation.parcellation.ParcellationStage attribute\), 139](#)
[create_workflow_custom\(\) \(cmp.stages.segmentation.segmentation.SegmentationStage attribute\), 141](#)
[CreateAcqpFile, 166](#)
[CreateBIDSStandardParcellationLabelIndexMappingFile, 142](#)
[CreateCMPParcellationNodeDescriptionFilesFromBIDSFile, 142](#)
[CreateIndexFile, 166](#)
[CreateMultipleCMPParcellationNodeDescriptionFilesFromBIDSFile, 143](#)
[crop_and_move_datasets\(\) \(in module cmtklib.parcellation\), 217](#)
[crop_and_move_WM_and_GM\(\) \(in module cmtklib.parcellation\), 216](#)
[crop_at_gm_wmi \(cmp.stages.diffusion.tracking.MRtrix_tracking_config attribute\), 107](#)
[CSD, 155](#)
[csf \(cmp.stages.functional.functionalMRI.FunctionalMRIConfig attribute\), 108](#)
[custom_aparcaseg \(cmp.stages.segmentation.segmentation.SegmentationConfig attribute\), 120](#)
[custom_aparcaseg_group \(cmp.bidsappmanager.stages.segmentation.segmentation.SegmentationConfigUI attribute\), 139](#)
[custom_brainmask \(cmp.stages.segmentation.segmentation.SegmentationConfig attribute\), 120](#)
[custom_brainmask_group \(cmp.bidsappmanager.stages.segmentation.segmentation.SegmentationConfigUI attribute\), 139](#)
[custom_csf_mask \(cmp.stages.segmentation.segmentation.SegmentationConfig attribute\), 120](#)
[custom_csf_mask_group \(cmp.bidsappmanager.stages.segmentation.segmentation.SegmentationConfigUI attribute\), 139](#)
[custom_gm_mask \(cmp.stages.segmentation.segmentation.SegmentationConfig attribute\), 120](#)
[custom_gm_mask_group \(cmp.bidsappmanager.stages.segmentation.segmentation.SegmentationConfigUI attribute\), 139](#)
[custom_parcellation \(cmp.bidsappmanager.stages.parcellation.parcellation.ParcellationConfig attribute\), 110](#)
[custom_parcellation_group \(cmp.bidsappmanager.stages.parcellation.parcellation.ParcellationConfigUI attribute\), 139](#)
[custom_track_file \(cmp.stages.diffusion.diffusion.DiffusionConfig attribute\), 100](#)
[custom_wm_mask \(cmp.stages.segmentation.segmentation.SegmentationConfig attribute\), 120](#)
[custom_wm_mask_group \(cmp.bidsappmanager.stages.segmentation.segmentation.SegmentationConfigUI attribute\), 139](#)
[CustomAparcasegBIDSFile \(class in cmtklib.bids.io\), 140](#)
[CustomBrainMaskBIDSFile \(class in cmtklib.bids.io\), 141](#)
[CustomCSFMaskBIDSFile \(class in cmtklib.bids.io\), 141](#)
[CustomGMMaskBIDSFile \(class in cmtklib.bids.io\), 141](#)
[CustomParcellationBIDSFile \(class in cmtklib.bids.io\), 142](#)
[CustomWMMaskBIDSFile \(class in cmtklib.bids.io\), 142](#)
[define_custom_mapping\(\) \(cmp.pipelines.anatomical.anatomical.AnatomicalPipeline method\), 90](#)
[define_custom_mapping\(\) \(cmp.pipelines.diffusion.diffusion.DiffusionPipeline method\), 92](#)
[define_custom_mapping\(\) \(cmp.pipelines.functional.fMRI.fMRIPipeline method\), 96](#)
[define_inspect_outputs\(\) \(cmp.stages.connectome.connectome.ConnectomeStage method\), 98](#)
[define_inspect_outputs\(\) \(cmp.stages.connectome.fmri_connectome.ConnectomeStage method\), 99](#)
[define_inspect_outputs\(\) \(cmp.stages.diffusion.diffusion.DiffusionStage method\), 102](#)
[define_inspect_outputs\(\) \(cmp.stages.functional.functionalMRI.FunctionalMRIStage method\), 100](#)
[define_inspect_outputs\(\) \(cmp.stages.parcellation.parcellation.ParcellationStage method\), 111](#)
[define_inspect_outputs\(\) \(cmp.stages.preprocessing.fmri_preprocessing.PreprocessingStage method\), 112](#)

method), 112

define_inspect_outputs() (cmp.stages.preprocessing.preprocessing.PreprocessingStageUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 133

define_inspect_outputs() (cmp.stages.registration.registration.RegistrationStageUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 133

define_inspect_outputs() (cmp.stages.segmentation.segmentation.SegmentationStageUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 133

denoising (cmp.stages.preprocessing.preprocessing.PreprocessingStageUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 133

denoising_algo (cmp.stages.preprocessing.preprocessing.PreprocessingStageUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 133

desc (cmr.klib.bids.io.CustomBIDSFile attribute), 141

description (cmp.stages.preprocessing.preprocessing.PreprocessingStageUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 133

desired_number_of_tracks (cmp.stages.diffusion.tracking.MRtrix_tracking_config attribute), 106

Despike, 145

despiking (cmp.stages.preprocessing.fmri_preprocessing.PreprocessingStageUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 133

Detrending, 207

DiffUnpack, 152

diffusion (cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipeline (class in cmp.bidsappmanager.pipelines.diffusion.diffusion), 91

diffusion_imaging_model (cmp.project.CMP_Project_Info attribute), 83

diffusion_imaging_model (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 100

diffusion_imaging_model (cmp.stages.registration.registration.RegistrationConfig attribute), 115

diffusion_imaging_model_editor (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 100

diffusion_model (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 101

diffusion_model_editor (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 101

DiffusionConfig (class in cmp.stages.diffusion.diffusion), 100

DiffusionConfigUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 133

DiffusionPipeline (class in cmp.pipelines.diffusion.diffusion), 91

DiffusionPipelineUI (class in cmp.bidsappmanager.pipelines.diffusion.diffusion), 128

DiffusionStage (class in cmp.stages.diffusion.diffusion), 101

DiffusionStageUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 133

Dilate_rois (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 100

dilation_kernel (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 100

dilation_radius (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 100

dmri_available (cmp.project.CMP_Project_Info attribute), 84

dmri_available_config (cmp.project.CMP_Project_Info attribute), 85

dmri_bids_acq (cmp.pipelines.diffusion.diffusion.Global_Configuration attribute), 94

dmri_bids_acq (cmp.project.CMP_Project_Info attribute), 83

dmri_bids_acqs (cmp.project.CMP_Project_Info attribute), 83

dmri_config_error_msg (cmp.project.CMP_Project_Info attribute), 85

dmri_config_to_load (cmp.project.CMP_Project_Info attribute), 85

dmri_custom_last_stage (cmp.project.CMP_Project_Info attribute), 85

dmri_inputs_checked (cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler attribute), 128

DirectionGetterTractography, 156

discard_n_volumes (cmp.stages.preprocessing.fmri_preprocessing.PreprocessingStageUI (class in cmp.bidsappmanager.stages.diffusion.diffusion), 133

Discard_tp, 208

attribute), 124
 dmri_inputs_checked
 (cmp.bidsappmanager.project.CMP_MainWindowHandler attribute), 125
 dmri_load_config_json() (in module cmtklib.config), 195
 dmri_pipeline(cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler attribute), 123
 dmri_pipeline(cmp.bidsappmanager.project.CMP_MainWindowHandler attribute), 125
 dmri_processed(cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler attribute), 124
 dmri_processed(cmp.bidsappmanager.project.CMP_MainWindowHandler attribute), 125
 dmri_run(cmp.project.CMP_Project_Info attribute), 84
 dmri_runs(cmp.project.CMP_Project_Info attribute), 84
 dmri_save_config() (in module cmtklib.config), 196
 dmri_stage_names(cmp.project.CMP_Project_Info attribute), 85
 docker_process(cmp.bidsappmanager.project.CMP_BIDSAppWindowHandler attribute), 122
 dof(cmp.stages.registration.registration.RegistrationConfig attribute), 117
 DTIEstimateResponseSH, 155
 DTIRecon, 151
 DTLUTGen, 147
 DVARs_thr(cmp.stages.connectome.fmri_connectome.ConnectomeConfig attribute), 98
 DWI2Tensor, 178
 DWIBiasCorrect, 178
 DWIDenoise, 179
E
 Eddy, 167
 eddy_correct_motion_correction
 (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 113
 eddy_correction_algo
 (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 113
 eddy_current_and_motion_correction
 (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 113
 EddyOpenMP, 168
 enabled(cmp.stages.common.Stage attribute), 122
 ENDC(cmtklib.util.BColors attribute), 218
 Erode, 180
 erode_mask() (in module cmtklib.parcellation), 217
 EstimateResponseForSH, 181
 extension(cmtklib.bids.io.CustomBIDSFile attribute), 141
 extract() (in module cmtklib.parcellation), 217
 extract_freesurfer_subject_dir() (in module cmtklib.util), 218
 extract_reconall_base_dir() (in module cmtklib.util), 218
 ExtractFSLGrad, 182
 ExtractHeaderVoxel2WorldMatrix, 173
 ExtractImageVoxelSize, 173
 ExtractMRTrixGrad, 183
 ExtractPDWFrom5TT, 201
F
 FQualityWindowHandler
 fa_thresh(cmp.stages.diffusion.tracking.Dipy_tracking_config attribute), 106
 FAIL(cmtklib.util.BColors attribute), 218
 fast_use_priors(cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 113
 FD_thr(cmp.stages.connectome.fmri_connectome.ConnectomeConfig attribute), 98
 fill_stages_outputs()
 (cmp.pipelines.common.Pipeline method), 8
 filter_fibers() (in module cmtklib.diffusion), 207
 FilterTractogram, 183
 flip_table_axis(cmp.bidsappmanager.stages.diffusion.reconstruction.Dipy_recon_config attribute), 134
 flip_table_axis(cmp.bidsappmanager.stages.diffusion.reconstruction.MRtrix_recon_config attribute), 134
 flip_table_axis(cmp.stages.diffusion.reconstruction.Dipy_recon_config attribute), 103
 flip_table_axis(cmp.stages.diffusion.reconstruction.MRtrix_recon_config attribute), 104
 FlipBvec, 202
 FlipTable, 203
 flirt_args(cmp.stages.registration.registration.RegistrationConfig attribute), 116
 fmri_available(cmp.project.CMP_Project_Info attribute), 84
 fmri_available_config
 (cmp.project.CMP_Project_Info attribute), 85
 fmri_config_error_msg
 (cmp.project.CMP_Project_Info attribute), 85
 fmri_config_to_load
 (cmp.project.CMP_Project_Info attribute), 85
 fmri_custom_last_stage
 (cmp.project.CMP_Project_Info attribute), 85
 fmri_inputs_checked
 (cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler attribute), 124
 fmri_inputs_checked
 (cmp.bidsappmanager.project.CMP_MainWindowHandler attribute), 124

`global_conf(cmp.pipelines.functional.fMRI.fMRIPipelineinit_dmri_project()` (in module `cmp.bidsappmanager.project`), 126
 attribute), 96
`Global_Configuration` (class in `init_dmri_project()` (in module `cmp.project`), 86
`cmp.pipelines.anatomical.anatomical`), 90
`Global_Configuration` (class in `init_fmri_project()` (in module `cmp.project`), 86
`cmp.pipelines.diffusion.diffusion`), 94
`Global_Configuration` (class in `input_folders(cmp.pipelines.anatomical.anatomical.AnatomicalPipeline`
`cmp.pipelines.functional.fMRI)`, 95
 attribute), 90
`global_nuisance(cmp.stages.functional.functionalMRI.FunctionalMRIStage` (in module `cmp.pipelines.diffusion.diffusion.DiffusionPipeline`
 attribute), 108
 attribute), 93
`gmwmi_seeding(cmp.stages.preprocessing.preprocessing.PreprocessingStage` (in module `cmp.pipelines.functional.fMRI.fMRIPipeline`
 attribute), 114
 attribute), 96
`gmwmi_seeding(cmp.stages.registration.registration.RegistrationStage` (in module `cmp.project`), 117
 attribute), 117
`group_analysis_sconn()` (in module `cmtk-lib.connectome`), 201
 attribute), 131
 inspect_output_button
 (cmp.bidsappmanager.stages.connectome.connectome.ConnectomeStageUI
 attribute), 132
 inspect_output_button
 (cmp.bidsappmanager.stages.connectome.fmri_connectome.Connec
 attribute), 132
 inspect_output_button
 (cmp.bidsappmanager.stages.diffusion.diffusion.DiffusionStageUI
 attribute), 133
`has_run()` (cmp.stages.connectome.connectome.ConnectomeStage (cmp.bidsappmanager.stages.diffusion.diffusion.DiffusionStageUI
 method), 98
 attribute), 133
`has_run()` (cmp.stages.connectome.fmri_connectome.ConnectomeStage inspect_output_button
 method), 99
 (cmp.bidsappmanager.stages.functional.functionalMRI.Functiona
 attribute), 135
`has_run()` (cmp.stages.diffusion.diffusion.DiffusionStage inspect_output_button
 method), 102
 (cmp.bidsappmanager.stages.parcellation.parcellation.Parcellatio
 attribute), 136
`has_run()` (cmp.stages.functional.functionalMRI.FunctionalMRIStage inspect_output_button
 method), 109
 (cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing.P
 attribute), 137
`has_run()` (cmp.stages.parcellation.parcellation.ParcellationStage inspect_output_button
 method), 111
 (cmp.bidsappmanager.stages.preprocessing.preprocessing.Prepro
 attribute), 138
`has_run()` (cmp.stages.preprocessing.fmri_preprocessing.PreprocessingStage inspect_output_button
 method), 112
 (cmp.bidsappmanager.stages.preprocessing.preprocessing.Prepro
 attribute), 138
`has_run()` (cmp.stages.preprocessing.preprocessing.PreprocessingStage inspect_output_button
 method), 114
 (cmp.bidsappmanager.stages.registration.registration.Registration
 attribute), 138
`has_run()` (cmp.stages.registration.registration.RegistrationStage inspect_output_button
 method), 118
 (cmp.bidsappmanager.stages.segmentation.segmentation.Segment
 attribute), 140
`has_run()` (cmp.stages.segmentation.segmentation.SegmentationStage inspect_output_button
 method), 121
 (cmp.bidsappmanager.stages.segmentation.segmentation.Segment
 attribute), 140
`HEADER(cmtklib.util.BColors attribute)`, 218
 inspect_outputs(cmp.stages.common.Stage attribute),
 122
`imaging_model(cmp.pipelines.functional.fMRI.Global_Configuration`
 attribute), 95
 inspect_outputs_enum(cmp.stages.common.Stage at-
 tribute), 122
`imaging_model(cmp.stages.diffusion.reconstruction.Dipy_recon_conf`
 attribute), 103
 inspect_outputs_view
 (cmp.bidsappmanager.stages.connectome.connectome.Connectom
 attribute), 131
`imaging_model(cmp.stages.diffusion.tracking.Dipy_tracking_conf`
 attribute), 105
 inspect_outputs_view
 (cmp.bidsappmanager.stages.connectome.fmri_connectome.Conn
 attribute), 132
`include_thalamic_nuclei_parcellation` (cmp.stages.parcellation.parcellation.ParcellationConfig
 attribute), 109
 inspect_outputs_view
 (cmp.bidsappmanager.stages.diffusion.diffusion.DiffusionStageUI
 attribute), 133
`init(cmp.stages.registration.registration.RegistrationConfig`
 attribute), 117
`init_anat_project()` (in module `cmp.bidsappmanager.project`), 126
 inspect_outputs_view
 (cmp.bidsappmanager.stages.functional.functionalMRI.Functiona
 attribute), 133
`init_anat_project()` (in module `cmp.project`), 86
 (cmp.bidsappmanager.stages.functional.functionalMRI.Functiona

attribute), 135
 inspect_outputs_view (cmp.bidsappmanager.stages.parcellation.parcellation.ParcellationConfig attribute), 136
 inspect_outputs_view (cmp.bidsappmanager.stages.preprocessing.fmri_fmri.PreprocessingConfig attribute), 137
 inspect_outputs_view (cmp.bidsappmanager.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 138
 inspect_outputs_view (cmp.bidsappmanager.stages.registration.registration.RegistrationConfig attribute), 139
 inspect_outputs_view (cmp.bidsappmanager.stages.segmentation.segmentation.SegmentationConfig attribute), 140
 interpolation (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 113
 is_running() (cmp.stages.common.Stage method), 122
 is_tool() (in module cmp.bidsappmanager.project), 126
 isavailable() (in module cmtklib.util), 219
 ismember() (cmtklib.parcellation.CombineParcellations method), 212
 isotropic_interpolation (cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 119
 isotropic_vox_size (cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 119
L
 label (cmtklib.bids.io.CustomBIDSFile attribute), 141
 laplacian_regularization (cmp.stages.diffusion.reconstruction.Dipy_recon_config attribute), 103
 laplacian_weighting (cmp.stages.diffusion.reconstruction.Dipy_recon_config attribute), 103
 launch_process() (cmp.pipelines.common.Pipeline method), 88
 length() (in module cmtklib.util), 219
 lmax_order (cmp.stages.diffusion.reconstruction.Dipy_recon_config attribute), 103
 lmax_order (cmp.stages.diffusion.reconstruction.MRtrix_recon_config attribute), 105
 load_anat_config_file() (cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler method), 124
 load_dataset() (cmp.bidsappmanager.project.CMP_MainWindowHandler method), 126
 load_dmri_config_file() (cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler method), 124
 load_fmri_config_file() (cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler method), 124
 load_graphs() (in module cmtklib.util), 220
 load_project() (cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler method), 124
 local_model (cmp.stages.diffusion.reconstruction.MRtrix_recon_config attribute), 103
 local_model_editor (cmp.stages.diffusion.reconstruction.Dipy_recon_config attribute), 103
 local_model_editor (cmp.stages.diffusion.reconstruction.MRtrix_recon_config attribute), 104
 log_visualization (cmp.bidsappmanager.stages.connectome.connectome.ConnectomeConfig attribute), 97
 log_visualization (cmp.bidsappmanager.stages.connectome.fmri_connectome.ConnectomeConfig attribute), 97
 log_visualization (cmp.stages.connectome.connectome.ConnectomeConfig attribute), 97
 log_visualization (cmp.stages.connectome.fmri_connectome.ConnectomeConfig attribute), 99
M
 magn() (in module cmtklib.util), 220
 make_isotropic (cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 119
 Make_Mrtrix_Seeds, 203
 Make_Seeds, 204
 manage_bidsapp_procs() (cmp.bidsappmanager.project.CMP_BIDSAppWindowHandler class method), 123
 MAPMRI, 158
 mapmri (cmp.stages.diffusion.reconstruction.Dipy_recon_config attribute), 103
 MathsCommand, 170
 max_angle (cmp.stages.diffusion.tracking.Dipy_tracking_config attribute), 106
 mean_curvature() (in module cmtklib.util), 220
 modalities (cmp.pipelines.diffusion.diffusion.Global_Configuration attribute), 94
 module
 cmp.bidsappmanager.pipelines.anatomical,
 cmp.bidsappmanager.pipelines.anatomical.anatomical,
 cmp.bidsappmanager.pipelines.diffusion,
 cmp.bidsappmanager.pipelines.diffusion.diffusion,
 cmp.bidsappmanager.pipelines.functional,
 cmp.bidsappmanager.pipelines.functional.eeg,

cmp.bidsappmanager.pipelines.functional.fMRI, 130
 cmp.bidsappmanager.project, 122
 cmp.bidsappmanager.stages, 131
 cmp.bidsappmanager.stages.connectome, 131
 cmp.bidsappmanager.stages.connectome.connectome, 131
 cmp.bidsappmanager.stages.connectome.fmri_connectome, 132
 cmp.bidsappmanager.stages.diffusion, 133
 cmp.bidsappmanager.stages.diffusion.diffusion, 133
 cmp.bidsappmanager.stages.diffusion.reconstruction, 134
 cmp.bidsappmanager.stages.diffusion.tracking, 134
 cmp.bidsappmanager.stages.functional, 135
 cmp.bidsappmanager.stages.functional.functionalMRI, 135
 cmp.bidsappmanager.stages.parcellation, 136
 cmp.bidsappmanager.stages.parcellation.parcellation, 136
 cmp.bidsappmanager.stages.preprocessing, 137
 cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing, 137
 cmp.bidsappmanager.stages.preprocessing.preprocessing, 137
 cmp.bidsappmanager.stages.registration, 138
 cmp.bidsappmanager.stages.registration.registration, 138
 cmp.bidsappmanager.stages.segmentation, 139
 cmp.bidsappmanager.stages.segmentation.segmentation, 139
 cmp.parser, 82
 cmp.pipelines.anatomical, 89
 cmp.pipelines.anatomical.anatomical, 89
 cmp.pipelines.common, 88
 cmp.pipelines.diffusion, 91
 cmp.pipelines.diffusion.diffusion, 91
 cmp.pipelines.functional, 94
 cmp.pipelines.functional.eeg, 94
 cmp.pipelines.functional.fMRI, 95
 cmp.project, 83
 cmp.stages, 97
 cmp.stages.common, 121
 cmp.stages.connectome, 97
 cmp.stages.connectome.connectome, 97
 cmp.stages.connectome.fmri_connectome, 98
 cmp.stages.diffusion, 100
 cmp.stages.diffusion.diffusion, 100
 cmp.stages.diffusion.reconstruction, 103
 cmp.stages.diffusion.tracking, 105
 cmp.stages.functional, 108
 cmp.stages.functional.functionalMRI, 108
 cmp.stages.parcellation, 109
 cmp.stages.parcellation.parcellation, 109
 cmp.stages.preprocessing, 111
 cmp.stages.preprocessing.fmri_preprocessing, 111
 cmp.stages.preprocessing.preprocessing, 112
 cmp.stages.registration, 115
 cmp.stages.registration.registration, 115
 cmp.stages.segmentation, 119
 cmp.stages.segmentation.segmentation, 119
 cmtklib, 140
 cmtklib.bids, 140
 cmtklib.bids.io, 140
 cmtklib.bids.utils, 142
 cmtklib.config, 194
 cmtklib.connectome, 198
 cmtklib.diffusion, 201
 cmtklib.functionalMRI, 207
 cmtklib.interfaces, 143
 cmtklib.interfaces.afni, 143
 cmtklib.interfaces.ants, 146
 cmtklib.interfaces.camino, 147
 cmtklib.interfaces.camino2trackvis, 150
 cmtklib.interfaces.diffusion_toolkit, 151
 cmtklib.interfaces.dipy, 155
 cmtklib.interfaces.freesurfer, 161
 cmtklib.interfaces.fsl, 164
 cmtklib.interfaces.misc, 173
 cmtklib.interfaces.mrtrix3, 174
 cmtklib.parcellation, 211
 cmtklib.util, 218
 motion(*cmp.stages.functional.functionalMRI.FunctionalMRIConfig*
 attribute), 108
 motion_correction(*cmp.stages.preprocessing.fmri_preprocessing.PreprocessingConfig*
 attribute), 111
 MRConvert, 185
 MRCrop, 187
 MRThreshold, 188
 MRTransform, 189
 MRTrir3Base, 190
 MRtrix_mul, 190
 MRtrix_recon_config (class in
 cmp.stages.diffusion.reconstruction), 104
 mrtrix_recon_config
 (*cmp.stages.diffusion.diffusion.DiffusionConfig*
 attribute), 101
 MRtrix_recon_configUI (class in
 cmp.bidsappmanager.stages.diffusion.reconstruction),
 134

MRtrix_tracking_config (class in [cmp.stages.diffusion.tracking](#)), 106

mrtrix_tracking_config (cmp.stages.diffusion.diffusion.DiffusionConfig attribute), 101

MRtrix_tracking_configUI (class in [cmp.bidsappmanager.stages.diffusion.tracking](#)), 134

MultipleANTsApplyTransforms, 146

N

new_project() (cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler method), 124

no_search (cmp.stages.registration.registration.RegistrationConfig attribute), 117

now (cmp.pipelines.anatomical.anatomical.AnatomicalPipeline attribute), 90

now (cmp.pipelines.diffusion.diffusion.DiffusionPipeline attribute), 93

now (cmp.pipelines.functional.fMRI.fMRIPipeline attribute), 96

Nuisance_regression, 208

number_of_cores (cmp.pipelines.common.Pipeline attribute), 88

number_of_cores (cmp.project.CMP_Project_Info attribute), 86

number_of_seeds (cmp.stages.diffusion.tracking.Dipy_tracking_info attribute), 106

number_of_threads (cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 120

O

OKBLUE (cmtklib.util.BColors attribute), 218

OKGREEN (cmtklib.util.BColors attribute), 218

ordered_stage_list (cmp.pipelines.anatomical.anatomical.AnatomicalPipeline attribute), 90

ordered_stage_list (cmp.pipelines.diffusion.diffusion.DiffusionPipeline attribute), 93

ordered_stage_list (cmp.pipelines.functional.fMRI.fMRIPipeline attribute), 96

Orient, 170

output_dir (cmp.stages.common.Stage attribute), 122

output_directory (cmp.project.CMP_Project_Info attribute), 83

output_types (cmp.bidsappmanager.stages.connectome.connectome.ConnectomeConfig attribute), 131

output_types (cmp.bidsappmanager.stages.connectome.fMRI_connectome.fMRI_connectome attribute), 132

output_types (cmp.stages.connectome.connectome.ConnectomeConfig attribute), 97

output_types (cmp.stages.connectome.fnri_connectome.fnri_connectome attribute), 99

P

Parcellate, 213

ParcellateBrainstemStructures, 214

ParcellateHippocampalSubfields, 214

ParcellateThalamus, 215

parcellation (cmp.bidsappmanager.pipelines.anatomical.anatomical.AnatomicalPipeline attribute), 127

parcellation_scheme (cmp.project.CMP_Project_Info attribute), 84

parcellation_scheme (cmp.stages.parcellation.parcellation.ParcellationConfig attribute), 109

parcellation_scheme_editor (cmp.stages.parcellation.parcellation.ParcellationConfig attribute), 109

ParcellationConfig (class in [cmp.stages.parcellation.parcellation](#)), 109

ParcellationConfigUI (class in [cmp.bidsappmanager.stages.parcellation.parcellation](#)), 136

ParcellationStage (class in [cmp.stages.parcellation.parcellation](#)), 110

ParcellationStageUI (class in [cmp.bidsappmanager.stages.parcellation.parcellation](#)), 136

partial_volume_estimation (cmp.stages.preprocessing.preprocessing.PreprocessingConfig attribute), 113

PicoPDFs, 148

Pipeline (class in [cmp.pipelines.common](#)), 88

pipeline (cmp.pipelines.common.ProcessThread attribute), 88

pipeline (cmp.stages.registration.registration.RegistrationConfig attribute), 115

pipeline_group (cmp.bidsappmanager.pipelines.anatomical.anatomical.AnatomicalPipeline attribute), 128

pipeline_group (cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipeline attribute), 129

pipeline_group (cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipeline attribute), 130

pipeline_mode (cmp.stages.parcellation.parcellation.ParcellationConfig attribute), 109

positivity_constraint (cmp.stages.diffusion.reconstruction.Dipy_recon_config attribute), 103

preprocessing (cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipeline attribute), 128

preprocessing (cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipeline attribute), 130

PreprocessingConfig (class in [cmp.stages.preprocessing.fnri_preprocessing](#)), 111

PreprocessingConfig (class in [cmp.stages.preprocessing](#)), 111

`cmp.stages.preprocessing.preprocessing)`,
[112](#)
PreprocessingConfigUI (class in `recon_mode` (`cmp.stages.diffusion.reconstruction.MRtrix_recon_config`
`cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing` attribute), [105](#)
[137](#)
PreprocessingConfigUI (class in `(cmp.stages.diffusion.diffusion.DiffusionConfig`
`cmp.bidsappmanager.stages.preprocessing.preprocessing)`, attribute), [100](#)
[137](#)
PreprocessingStage (class in `(cmp.stages.diffusion.diffusion.DiffusionConfig`
`cmp.stages.preprocessing.fmri_preprocessing)`, attribute), [100](#)
[112](#)
PreprocessingStage (class in `cmp.bidsappmanager.project`), [126](#)
`cmp.stages.preprocessing.preprocessing)`,
[114](#)
PreprocessingStageUI (class in `registration` (`cmp.bidsappmanager.pipelines.diffusion.diffusion.Diffusion`
`cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing` attribute), [128](#)
[137](#)
PreprocessingStageUI (class in `registration_mode` (`cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipel`
`cmp.bidsappmanager.stages.preprocessing.preprocessing)`, attribute), [130](#)
[138](#)
PreprocessingStageUI (class in `registration_mode` (`cmp.stages.registration.registration.RegistrationCon`
`cmp.bidsappmanager.stages.preprocessing.preprocessing)`, attribute), [115](#)
[138](#)
print_blue() (in module `cmtklib.util`), [221](#)
print_error() (in module `cmtklib.util`), [221](#)
print_warning() (in module `cmtklib.util`), [221](#)
process() (`cmp.pipelines.anatomical.anatomical.AnatomicalPipeline`
`cmp.stages.registration.registration)`, [115](#)
method), [90](#)
process() (`cmp.pipelines.diffusion.diffusion.DiffusionPipeline`
`cmp.bidsappmanager.stages.registration.registration)`,
method), [93](#)
process() (`cmp.pipelines.functional.fMRI.fMRIPipeline`
`cmp.stages.registration.registration)`, [118](#)
method), [96](#)
process_type (`cmp.pipelines.anatomical.anatomical.Global_Configuration`
`cmp.bidsappmanager.stages.registration.registration)`,
attribute), [91](#)
process_type (`cmp.pipelines.diffusion.diffusion.Global_Configuration`
`cmp.bidsappmanager.stages.registration.registration)`,
attribute), [94](#)
process_type (`cmp.pipelines.functional.fMRI.Global_Configuration`
`cmp.bidsappmanager.stages.registration.registration)`,
attribute), [95](#)
processing_tool_editor
(`cmp.stages.diffusion.diffusion.DiffusionConfig`
attribute), [100](#)
ProcessThread (class in `cmp.pipelines.common`), [88](#)
ProgressThread (class in `cmp.pipelines.common`), [88](#)
ProgressWindow (class in `cmp.pipelines.common`), [89](#)
project_loaded (`cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler`
attribute), [123](#)
project_loaded (`cmp.bidsappmanager.project.CMP_MainWindowHandler`
attribute), [125](#)
pw (`cmp.pipelines.common.ProgressThread` attribute), [89](#)

R

radial_order (`cmp.stages.diffusion.reconstruction.Dipy_recon_config`
attribute), [103](#)
radial_order_values
(`cmp.stages.diffusion.reconstruction.Dipy_recon_config`
attribute), [104](#)

S

recon_mode (`cmp.stages.diffusion.reconstruction.Dipy_recon_config`
attribute), [103](#)
recon_mode (`cmp.stages.diffusion.reconstruction.MRtrix_recon_config`
attribute), [105](#)
recon_processing_tool
(`cmp.stages.diffusion.diffusion.DiffusionConfig`
attribute), [100](#)
recon_processing_tool_editor
(`cmp.stages.diffusion.diffusion.DiffusionConfig`
attribute), [100](#)
refresh_folder() (in `module`
`cmp.bidsappmanager.project`), [126](#)
refresh_folder() (in module `cmp.project`), [87](#)
registration (`cmp.bidsappmanager.pipelines.diffusion.diffusion.Diffusion`
`cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipel`
attribute), [130](#)
registration (`cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipel`
attribute), [130](#)
registration_mode (`cmp.stages.registration.registration.RegistrationCon`
attribute), [115](#)
registration_mode_trait
(`cmp.stages.registration.registration.RegistrationConfig`
attribute), [115](#)
RegistrationConfig (class in
`cmp.stages.registration.registration)`, [115](#)
RegistrationConfigUI (class in
`cmp.bidsappmanager.stages.registration.registration)`,
[138](#)
RegistrationStage (class in
`cmp.stages.registration.registration)`, [118](#)
RegistrationStageUI (class in
`cmp.bidsappmanager.stages.registration.registration)`,
[118](#)
repetition_time (`cmp.stages.preprocessing.fmri_preprocessing.Preproce`
attribute), [111](#)
res (`cmtklib.bids.io.CustomBIDSFile` attribute), [141](#)
resampling (`cmp.stages.preprocessing.preprocessing.PreprocessingConfig`
attribute), [113](#)
return_button_style_sheet() (in module `cmtk-`
`lib.util`), [221](#)
ROI_idx (`cmtklib.diffusion.Make_Mrtrix_Seeds` at-
tribute), [204](#)
ROI_idx (`cmtklib.diffusion.Make_Seeds` attribute), [205](#)
run() (`cmp.pipelines.common.ProcessThread` method),
[88](#)
run() (`cmp.pipelines.common.ProgressThread` method),
[89](#)
run_individual() (in module `cmp.project`), [87](#)

[save_fmri_config_file\(\)](#) (*cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler* class method), 124
[save_fibers\(\)](#) (in module *cmtklib.connectome*), 201
[save_fmri_config_file\(\)](#) (*cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler* class method), 124
[Scrubbing](#), 210
[SD](#) (*cmp.stages.diffusion.tracking.Dipy_tracking_config* attribute), 105
[SD](#) (*cmp.stages.diffusion.tracking.MRtrix_tracking_config* attribute), 106
[seed_density](#) (*cmp.stages.diffusion.tracking.Dipy_tracking_config* attribute), 106
[seed_from_gmwm](#) (*cmp.stages.diffusion.tracking.Dipy_tracking_config* attribute), 106
[seed_from_gmwm](#) (*cmp.stages.diffusion.tracking.MRtrix_tracking_config* attribute), 107
[seg_tool](#) (*cmp.stages.segmentation.segmentation.SegmentationConfig* attribute), 119
[segment_brainstem](#) (*cmp.stages.parcellation.parcellation.ParcellationConfig* attribute), 110
[segment_hippocampal_subfields](#) (*cmp.stages.parcellation.parcellation.ParcellationConfig* attribute), 110
[segmentation](#) (*cmp.bidsappmanager.pipelines.anatomical.anatomical.Global_Configuration* attribute), 127
[SegmentationConfig](#) (class in *cmp.stages.segmentation.segmentation*), 119
[SegmentationConfigUI](#) (class in *cmp.bidsappmanager.stages.segmentation.segmentation*), 139
[SegmentationStage](#) (class in *cmp.stages.segmentation.segmentation*), 120
[SegmentationStageUI](#) (class in *cmp.bidsappmanager.stages.segmentation.segmentation*), 140
[set_pipeline_attributes_from_config\(\)](#) (in module *cmtklib.config*), 197
[sh_order](#) (*cmp.stages.diffusion.tracking.Dipy_tracking_config* attribute), 106
[SHORE](#), 159
[shore_constrain_e0](#) (*cmp.stages.diffusion.reconstruction.Dipy_recon_config* attribute), 104
[shore_lambda_l](#) (*cmp.stages.diffusion.reconstruction.Dipy_recon_config* attribute), 104
[shore_lambda_n](#) (*cmp.stages.diffusion.reconstruction.Dipy_recon_config* attribute), 104
[shore_positive_constraint](#) (*cmp.stages.diffusion.reconstruction.Dipy_recon_config* attribute), 104
[shore_radial_order](#) (*cmp.stages.diffusion.reconstruction.Dipy_recon_config* attribute), 104
[shore_zeta](#) (*cmp.stages.diffusion.reconstruction.Dipy_recon_config* attribute), 104
[show_bids_app_window\(\)](#) (*cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler* class method), 124
[sift](#) (*cmp.stages.diffusion.tracking.MRtrix_tracking_config* attribute), 107
[SIFT2](#), 191
[single_fib_thr](#) (*cmp.stages.diffusion.reconstruction.Dipy_recon_config* attribute), 103
[single_fib_thr](#) (*cmp.stages.diffusion.reconstruction.MRtrix_recon_config* attribute), 105
[slice_timing](#) (*cmp.stages.preprocessing.fmri_preprocessing.PreprocessingConfig* attribute), 111
[small_delta](#) (*cmp.stages.diffusion.reconstruction.Dipy_recon_config* attribute), 104
[SplitDiffusion](#), 205
[StageNamesConfig](#) (*cmp.stages.common*), 121
[stage_names](#) (*cmp.pipelines.common.ProgressThread* attribute), 89
[stages](#) (*cmp.pipelines.common.ProgressThread* attribute), 89
[start_bids_app\(\)](#) (*cmp.bidsappmanager.project.CMP_BIDSAppWindowHandler* method), 123
[start_bids_app_process\(\)](#) (*cmp.bidsappmanager.project.CMP_BIDSAppWindowHandler* class method), 123
[step_size](#) (*cmp.stages.diffusion.tracking.Dipy_tracking_config* attribute), 106
[stop_bids_app\(\)](#) (*cmp.bidsappmanager.project.CMP_BIDSAppWindowHandler* class method), 123
[StreamlineTrack](#), 191
[strip_suffix\(\)](#) (in module *cmp.stages.diffusion.diffusion*), 102
[subject](#) (*cmp.pipelines.anatomical.anatomical.Global_Configuration* attribute), 91
[subject](#) (*cmp.pipelines.common.Pipeline* attribute), 88
[subject](#) (*cmp.pipelines.diffusion.diffusion.Global_Configuration* attribute), 94
[subject](#) (*cmp.project.CMP_Project_Info* attribute), 83
[subject](#) (*cmp.stages.connectome.connectome.ConnectomeConfig* attribute), 97
[subject](#) (*cmp.stages.connectome.fmri_connectome.ConnectomeConfig* attribute), 99
[subject_session](#) (*cmp.pipelines.anatomical.anatomical.Global_Configuration* attribute), 91
[subject_session](#) (*cmp.pipelines.diffusion.diffusion.Global_Configuration* attribute), 94
[subject_session](#) (*cmp.project.CMP_Project_Info* attribute), 83
[subject_sessions](#) (*cmp.project.CMP_Project_Info* attribute), 83

`subjects(cmp.pipelines.anatomical.anatomical.Global_Configuration_attribute)`, 91

`subjects(cmp.pipelines.diffusion.diffusion.Global_Configuration_attribute)`, 94

`subjects(cmp.project.CMP_Project_Info_attribute)`, 83

`suffix(cmtklib.bids.io.CustomBIDSFile_attribute)`, 140

T

`t1_available(cmp.project.CMP_Project_Info_attribute)`, 84

`Tck2Trk`, 205

`Tensor2Vector`, 193

`TensorInformedEudXTractography`, 160

`Tkregister2`, 162

`toolbox_derivatives_dir(cmtk-lib.bids.io.CustomBIDSFile_attribute)`, 140

`total_readout(cmp.stages.preprocessing.preprocessing.Runicodes2strConfig_module cmtklib.util)`, 221

`tracking_mode(cmp.stages.diffusion.tracking.Dipy_tracking_config cmp.bidsappmanager.project)`, 127

`tracking_mode(cmp.stages.diffusion.tracking.MRtrix_tracking_config cmp.project)`, 87

`tracking_processing_tool(cmp.stages.diffusion.diffusion.DiffusionConfig_attribute)`, 100

`tracking_processing_tool(cmp.stages.diffusion.reconstruction.Dipy_recon_config_attribute)`, 103

`tracking_processing_tool_editor(cmp.stages.diffusion.diffusion.DiffusionConfig_attribute)`, 100

`tracking_tool(cmp.stages.preprocessing.preprocessing.PreprocessingConfig_attribute)`, 113

`tracking_tool(cmp.stages.registration.registration.RegistrationConfig_attribute)`, 117

`traits_view(cmp.bidsappmanager.pipelines.anatomical.anatomical.PipelineUI_attribute)`, 128

`traits_view(cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipelineUI_attribute)`, 129

`traits_view(cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipeline_pipeline)`, 130

`traits_view(cmp.bidsappmanager.stages.connectome.connectome.ConnectomeConfigUI_attribute)`, 131

`traits_view(cmp.bidsappmanager.stages.connectome.fmri_connectome_fmriConnectomeConfigUI_attribute)`, 132

`traits_view(cmp.bidsappmanager.stages.diffusion.diffusion.DiffusionDiffusionPipeline_method)`, 133

`traits_view(cmp.bidsappmanager.stages.diffusion.reconstruction.ReconstructionDiffusionPipeline_method)`, 134

`traits_view(cmp.bidsappmanager.stages.diffusion.reconstruction.MRtrixRecon_configUI_attribute)`, 134

`traits_view(cmp.bidsappmanager.stages.diffusion.tracking.Dipy_tracking_diffusion.DiffusionPipeline_method)`, 134

`traits_view(cmp.bidsappmanager.stages.diffusion.tracking.MRtrixTracking_configUI_attribute)`, 134

`traits_view(cmp.bidsappmanager.stages.functionals.FunctionalDiffusionPipeline_method)`, 136

`traits_view(cmp.bidsappmanager.stages.preprocessing.fmri_preprocessing_fmriPreprocessingAttribute_attribute)`, 137

`traits_view(cmp.bidsappmanager.stages.preprocessing.preprocessing.PreprocessingPipeline_attribute)`, 137

`traits_view(cmp.bidsappmanager.stages.registration.registration.RegistrationRegistrationAttribute_attribute)`, 138

`traits_view(cmp.bidsappmanager.stages.segmentation.segmentation.SegmentationSegmentationAttribute_attribute)`, 139

U

`UNDERLINE(cmtklib.util.BColors_attribute)`, 218

`update_anat_last_processed()(in module cmp.bidsappmanager.project)`, 127

`update_anat_last_processed()(in module cmp.project)`, 87

`update_dipy_tracking_SD()(cmp.stages.diffusion.diffusion.DiffusionConfig_method)`, 101

`update_dipy_tracking_sh_order()(cmp.stages.diffusion.diffusion.DiffusionConfig_method)`, 101

`update_dmri_last_processed()(in module cmp.bidsappmanager.project)`, 127

`update_dmri_last_processed()(in module cmp.project)`, 87

`update_fmri_last_processed()(in module cmp.bidsappmanager.project)`, 127

`update_fmri_last_processed()(in module cmp.project)`, 87

`update_outputs_reConfgUI(cmp.pipelines.diffusion.diffusion.DiffusionPipeline_method)`, 93

`update_preprocessing_gmwmrmi()(cmp.pipelines.diffusion.diffusion.DiffusionPipeline_method)`, 93

[update_registration\(\)](#)
(cmp.pipelines.functional.fMRI.fMRIPipeline method), 97
[update_scrubbing\(\)](#) *(cmp.pipelines.functional.fMRI.fMRIPipeline method)*, 97
[update_subject_anat_pipeline\(\)](#)
(cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler method), 125
[update_subject_dmri_pipeline\(\)](#)
(cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler method), 125
[update_subject_fmri_pipeline\(\)](#)
(cmp.bidsappmanager.project.CMP_ConfigQualityWindowHandler method), 125
[update_tracking_tool\(\)](#)
(cmp.pipelines.diffusion.diffusion.DiffusionPipeline method), 93
[update_vizualization_layout\(\)](#)
(cmp.pipelines.diffusion.diffusion.DiffusionPipeline method), 93
[update_vizualization_logscale\(\)](#)
(cmp.pipelines.diffusion.diffusion.DiffusionPipeline method), 94
[UpdateGMWMInterfaceSeeding](#), 206
[use_act](#) *(cmp.stages.diffusion.tracking.Dipy_tracking_config attribute)*, 106
[use_act](#) *(cmp.stages.diffusion.tracking.MRtrix_tracking_config attribute)*, 107
[use_existing_freesurfer_data](#)
(cmp.stages.segmentation.segmentation.SegmentationConfig attribute), 119
[use_float_precision](#)
(cmp.stages.registration.registration.RegistrationConfig attribute), 115
[use_fsl_brain_mask](#) *(cmp.stages.segmentation.segmentation.SegmentationConfig attribute)*, 119
[uses_qform](#) *(cmp.stages.registration.registration.RegistrationConfig attribute)*, 117

V

[view_mode](#) *(cmp.bidsappmanager.pipelines.anatomical.anatomical.AnatomicalPipelineUI attribute)*, 128
[view_mode](#) *(cmp.bidsappmanager.pipelines.diffusion.diffusion.DiffusionPipelineUI attribute)*, 129
[view_mode](#) *(cmp.bidsappmanager.pipelines.functional.fMRI.fMRIPipelineUI attribute)*, 130
[Voxel2Image](#), 149

W

[WARNING](#) *(cmtklib.util.BColors attribute)*, 218
[wm](#) *(cmp.stages.functional.functionalMRI.FunctionalMRIConfig attribute)*, 108
[write_derivative_description\(\)](#) *(in module cmklib.bids.utils)*, 143