ElikoPy

Quentin Dessain, Mathieu Simon

Jul 28, 2021

GETTING STARTED

1 Introduction to ElikoPy **3**
- 1.1 Why use ElikoPy? . . . 3
- 1.2 Features . . . 3

2 Installation **5**
- 2.1 Manual Installation Steps . . . 5
- 2.2 Container Installation Steps . . . 7
- 2.3 Using ElikoPy on the CECI Cluster . . . 7

3 Project as an easy way to manage a study **9**
- 3.1 Main available processing steps . . . 9
- 3.2 Folder structure . . . 10

4 Typical usage for processing a study **13**
- 4.1 Header and initialisation . . . 13
- 4.2 Preprocessing . . . 14
- 4.3 whitematter mask . . . 14
- 4.4 Microstructural metrics computation . . . 15
- 4.5 Statistical Analysis . . . 15
- 4.6 Data Exportation . . . 15
- 4.7 Other parameters commonly available . . . 16

5 Preprocessing of diffusion images **17**
- 5.1 Reslice . . . 17
- 5.2 Brain Extraction . . . 18
- 5.3 MPPCA Denoising . . . 19
- 5.4 Gibbs Ringing Correction . . . 19
- 5.5 Susceptibility field estimation . . . 20
- 5.6 Eddy and motion correction . . . 21
- 5.7 Bias Field Correction . . . 22
- 5.8 Report . . . 22

6 T1 Preprocessing **23**

7 Project as an easy way to manage a study **25**

8 Project as an easy way to manage a study **27**

9 Project as an easy way to manage a study **29**

10 Examples **31**

11 Contributing **33**
11.1 Raise an Issue . . . 33
11.2 Write Documentation . . . 33
11.3 Contribute to the code . . . 33

12 elikopy package **37**
12.1 Submodules . . . 37
12.2 elikopy.core module . . . 37
12.3 elikopy.individual_subject_processing module . . . 46
12.4 elikopy.utils module . . . 49
12.5 Module contents . . . 53

13 License **55**

Python Module Index **69**

Index **71**

ElikoPy is Python library aiming at easing the processing of diffusion imaging for microstructural analysis. This Python library is based on

- DIPY, a python library for the analysis of MR diffusion imaging.
- Microstructure fingerprinting, a python library doing estimation of white matter microstructural properties from a dictionary of Monte Carlo diffusion MRI fingerprints.
- FSL, a comprehensive library of analysis tools for FMRI, MRI and DTI brain imaging data.
- DIAMOND, a c software that is characterizing brain tissue by assessment of the distribution of anisotropic microstructural environments in diffusion-compartment imaging.
- Dmipy, a python library estimating diffusion MRI-based microstructure features, used to fit and recover the parameters of multi-compartment microstructure models

This guide aims to give an introduction to ElikoPy and a brief installation instructions.

CHAPTER

ONE

INTRODUCTION TO ELIKOPY

ElikoPy is Python library aiming at easing the processing of diffusion imaging for microstructural analysis. ElikoPy expands state of the art pipeline frameworks by providing a complete quality assessment and quality reports for each processed subject, providing a standardized framework to ensure reproducibility and consistency, reducing error propagation and improve sensitivity. It also grants the possibility for clinicians to perform fast preprocessing for a large variety of studies with minimal knowledge.

The ElikoPy library was developed during a Master's thesis.

Note: If you wish to learn more about the library and its validation, we invite you to read our Master's thesis.

1.1 Why use ElikoPy?

Diffusion weighted magnetic resonance imaging (DW-MRI) is a rapidly evolving, non radiating and non invasive technique that allows to capture information on the brain microstructure through the restricted diffusion of water molecules. DW-MRI has seen a growing interest in the recent years motivating the acquisition of large multi-scanner multi-site data sets. The substantial acquisition time of this type of MRI sequence has also encouraged the extensive use of Echo Planar Imaging which suffers from additional artifacts and noise. Several tools have been developed in order to correct those individual problems but they come with the disadvantages of processing only one subject at a time and requiring different softwares making them cumbersome to use. This work presents and evaluates the performances of the ElikoPy library, a complete diffusion MRI processing pipeline that reduces common sources of artifact and captures information on the brain microstructure through multiple microstructural diffusion models. ElikoPy has been designed to deal with large databases and to be robust to different types of acquisitions.

1.2 Features

1. Preprocessing of your dMRI data.
2. Generation of a synthesized b0 for diffusion distortion correction using the based on the Synb0-DISCO repository. This synthesized b0 is usefull for topup if the acquistion was only performed with one phase encoding direction.
3. The library can compute the DTI, Noddi, DIAMOND and the novel Microstucturefingerprinting metric.
4. Complete quality reports to review each step of the processing.
5. Tissue segmentation from T1 images.
6. Ability to run subject and group wise statistics on the dataset.

CHAPTER

TWO

INSTALLATION

2.1 Manual Installation Steps

You will need a Linux system (CentOS 7 is recommended) to run ElikoPy and all its dependencies natively using a manual installation. Doing a manual installation is not recommended if you have only a limited knownledge in computer science.

2.1.1 Installation of the dependencies

You must first install dependency to your system. Some dependencies are optionnal while other are mandatory.

FSL installation (mandatory)

FSL is a mandatory comprehensive library dependency used among other steps for the preprocessing of diffusion images. FSL is available ready to run on the official FSL installation page.

FreeSurfer installation (optionnal)

FreeSurfer is a software package for the analysis and visualization of structural and functional neuroimaging data from cross-sectional or longitudinal studies. This software is mandatory when correcting from susceptibility distortion using T1 structural images in the preprocessing. To install it, visit the FreeSurfer Downloads page and pick a package archive suitable to the environment you are in.

ANTs installation (optionnal)

ANTs computes high-dimensional mappings to capture the statistics of brain structure and function. This software is mandatory when correcting from susceptibility distortion using T1 structural images in the preprocessing. ANTs can be compiled from source or installed via pre-built package using their Github page.

C3D installation (optionnal)

C3D is a command-line tool for converting 3D images between common file formats. The tool also includes a growing list of commands for image manipulation, such as thresholding and resampling. This software is mandatory when correcting from susceptibility distortion using T1 structural images in the preprocessing. A precompiled version of C3D is availabe on Sourceforge.

Microstructure Fingerprinting installation (recommended) (optionnal)

Microstructure Fingerprinting estimate the white matter microstructural properties from a dictionary of Monte Carlo diffusion MRI fingerprints. To install it, first download a copy of the MF repository.

```
git clone git@github.com:rensonnetg/microstructure_fingerprinting.git
```

Then, navigate to the folder where this repository was cloned or downloaded (the folder containing the setup.py file) and install the package as follows.

```
cd microstructure_fingerprinting
python setup.py install --user
```

DIAMOND installation (optionnal)

Unfortunatly, the DIAMOND code is not publically available. If you do not have it in your possesion, you will not be able to use this algorithm. If you have it, simply add the executable to your path.

2.1.2 Installation of ElikoPy

ElikoPy requires Python v3.7+ to run.To install it, first download a copy or clone the ElikoPy repository.

```
git clone git@github.com:Hyedryn/elikopy.git
```

After cloning the repo, you can either firstly install all the python dependencies including optionnal dependency used to speed up the code.

```
pip install -r requirements.txt --user
```

Or you can install directly the library with only the mandatory dependencies (if you performed the previous step, you still need to perform this step).

```
python3 setup.py install --user
```

Note: When using ElikoPy, do not forget to reference it among all of the used dependencies.

2.2 Container Installation Steps

To ease the installation of ElikoPy, a Singularity container is provided in the ElikoPy repository. To learn more about Singularity, you can visit their official website.

```
git clone https://github.com/Hyedryn/elikopy.git
cd /path/to/repo
sudo singularity build /path/to/elikopy.sif Singularity_elikopy
```

After building the container, ElikoPy can be run using the following command:

```
singularity run -e --contain
-B /path/to/study/directory/:/PROJECTS
-B /tmp:/tmp
-B /path/to/freesurfer/license.txt:/Software/freesurfer/license.txt
-B /path/to/cuda:/usr/local/cuda
--nv
/path/to/elikopy.sif
/path/to/script.py
```

The script.py file contains the Python code that will be executed inside the container. The path to the root directory in your python code must always be "/PROJECTS/" due to the folder binding.

Note: Binding the freesurfer license is optional and is only needed for Synb0-DisCo.

Note: Binding the cuda path is optional and is only needed to speed-up Synb0-DisCo or perform inter slice motion correction with Eddy FSL.

2.3 Using ElikoPy on the CECI Cluster

UCLouvain student who wish to use ElikoPy on the CECI cluster can use the existing installation present in the pilab project directory. First, the following line needs to be added to our *.bash_profile*.

```
source /CECI/proj/pilab/Software/config_elikopy.bash
```

Then, execute the following line of code to install ElikoPy:

```
source /CECI/proj/pilab/Software/install_elikopy.bash
```

If you wish to update your ElikoPy installation, you just need to execute again the preceding line of code.

Authorized user can update the local ElikoPy repository present in the PiLab directory using the following script. The local repository is update using the master branch of the remote Github repository.

```
source /CECI/proj/pilab/Software/update_elikopy.bash
```

These steps should be sufficient for the *lemaitre3* and *manneback* clusters. When using other clusters, some additional modules may need to be loaded (see the related CECI documentation for more information). We also strongly recommend you to familiarize yourself with slurm job when using ElikoPy on the CECI cluster.

CHAPTER

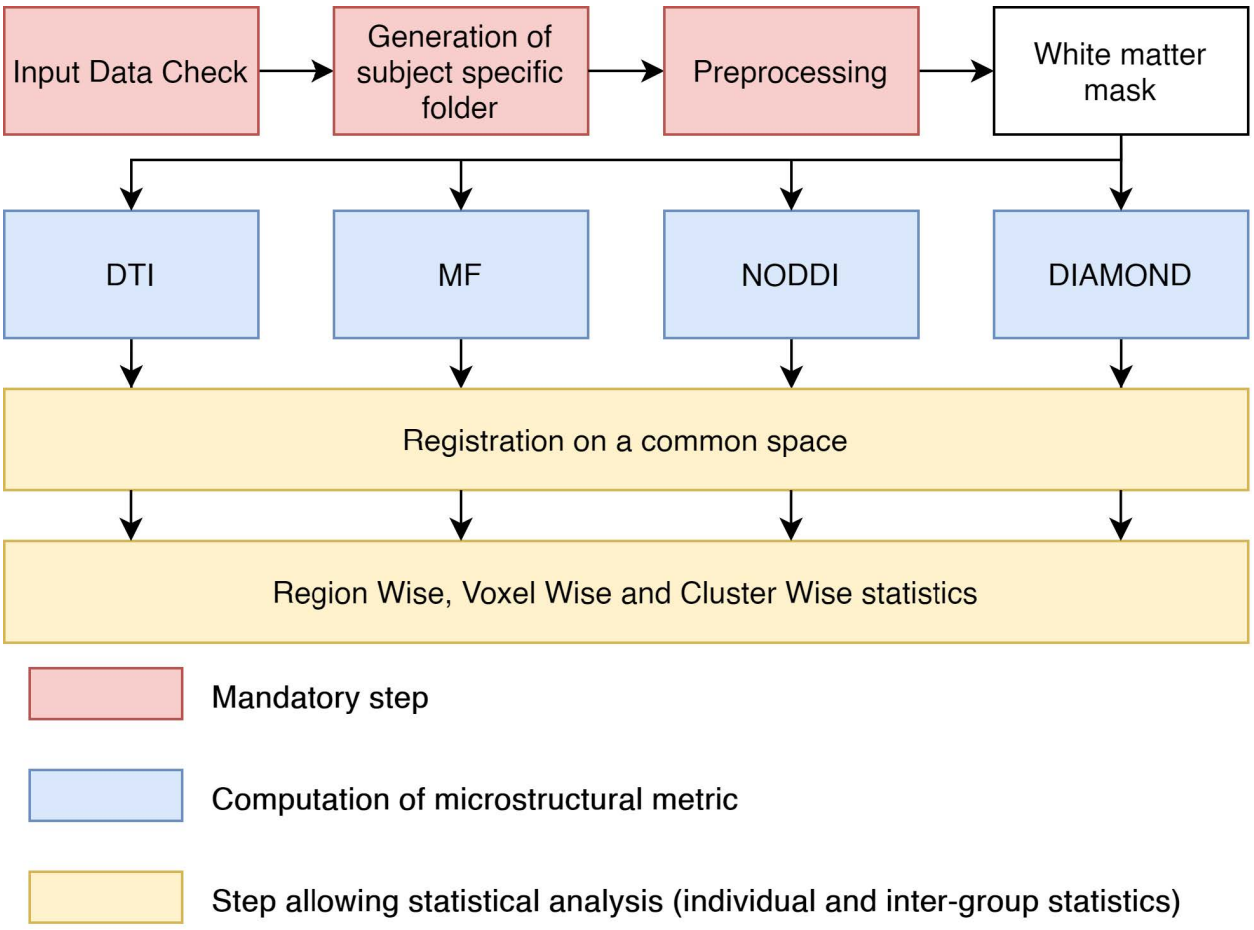
THREE

PROJECT AS AN EASY WAY TO MANAGE A STUDY

3.1 Main available processing steps

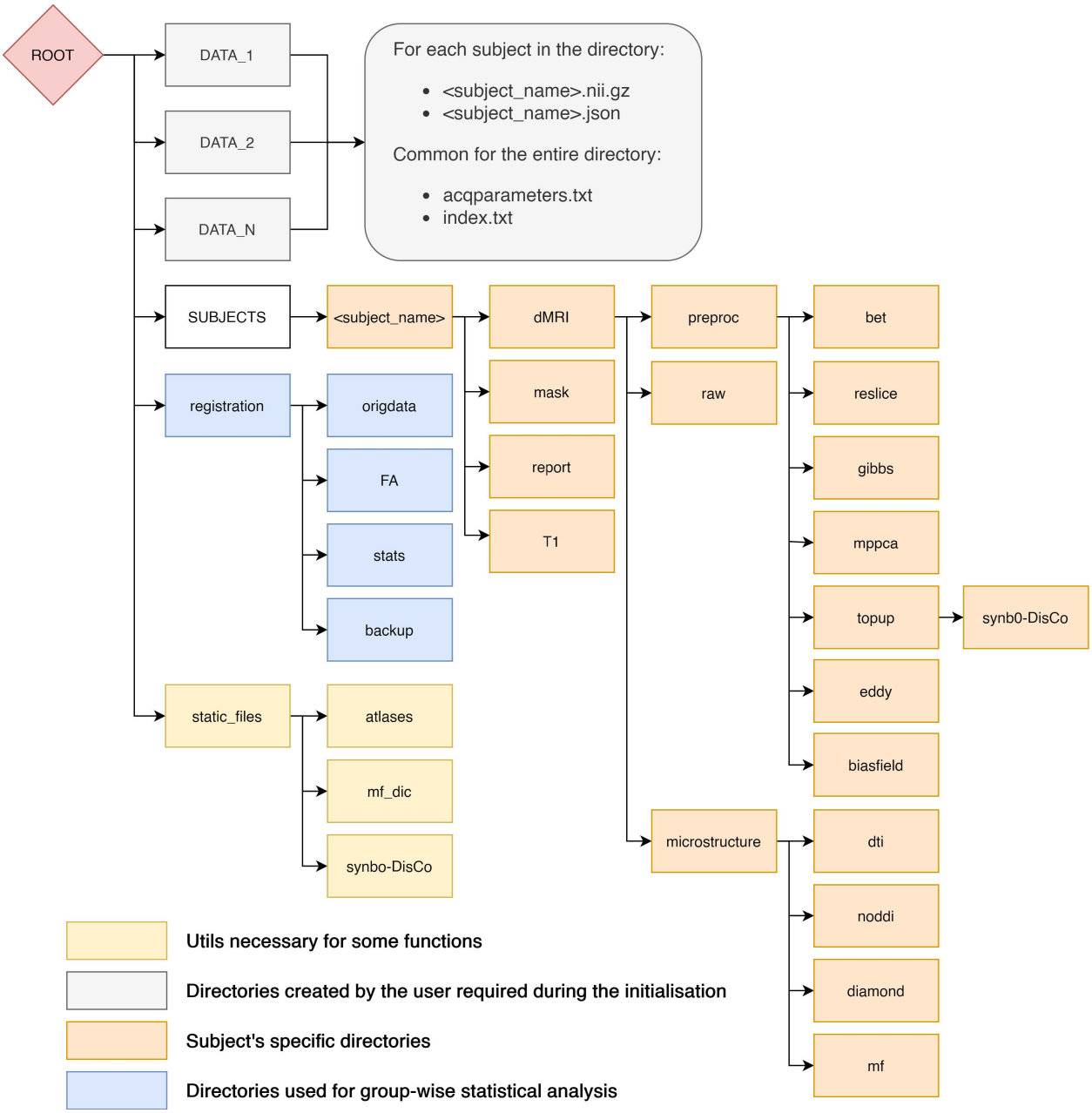
Using a rich set of functions, data processed with ElikoPy are firstly audited to ensure that the image dimension match the dimension of the b-value and b-vector files along the acquisitions parameters and index files. If successful, a dedicated storage folder for each subject is generated. Afterward, preprocessing can be applied to correct and enhanced the raw image. This includes, brain extraction, reslicing, Principal Component Analysis (PCA) based denoising using the Marchenko-Pastur distribution, suppression in image space of Gibbs ringing artifacts, estimation and correction of the susceptibility induced field, modeling and correction of subject movements and finally, bias field correction. After the preprocessing, a white matter mask registered on diffusion data can be obtained using a T1 image or computed directly from diffusion data by segmenting an Anisotropic Power (AP) map.

As seen in the figure below, four distinct algorithms for the estimation of the microstructure are available. Subsequently, the pipeline adds the possibility to register the computed metrics in a common space given by either one of the study subjects or by an atlas. Finally, from there, ElikoPy can output statistical results for population studies by aligning metrics from multiple subjects into a common space and performing region wise, voxel wise and cluster wise statistics.



3.2 Folder structure

The Elikopy toolbox follows a specific folder structure that prevents ambiguity and data losses when dealing with a large amount of subjects. The figure below illustrates the folder tree in ElikoPy.



The first type of folder present at the root of the Elikopy project are **data** folders. These folders contain all raw subject files belonging to the same class along with the associated acquisition parameters and index files. Classes are used to separate subjects that have different acquisition parameters or subjects that need to be separated from others groups of subjects. The pipeline does not have a limitation on the number of classes.

The **subjects** folder contains a subfolder for each valid subject presents in **data** folders. Along these subfolders, three json files are present. The subj_error.json file contains the list of subjects with invalid raw data, The subj_list.json file contains the list of valid subjects and the subj_type.json is a dictionary that maps each subjects subfolder to its data class.

Each subdirectory of the **subjects** folder contains the output of every preprocessing and processing function executed on the patient associated with the subdirectory. The output consists of NIfTI files, log files and some others files related to the specific functions.

The **registration** folder contains diffusion metrics registered to a common space, group wise statistics and voxel wise

statistics for each registered metric.

Finally, the **static_files** folder contains files mandatory for some processing steps of the library such as MF dictionary and Synb0-DisCo atlases.

CHAPTER

FOUR

TYPICAL USAGE FOR PROCESSING A STUDY

On this page is presented a basic usage of the ElikoPy library. More information on all these functions are available in the detailed guide.

4.1 Header and initialisation

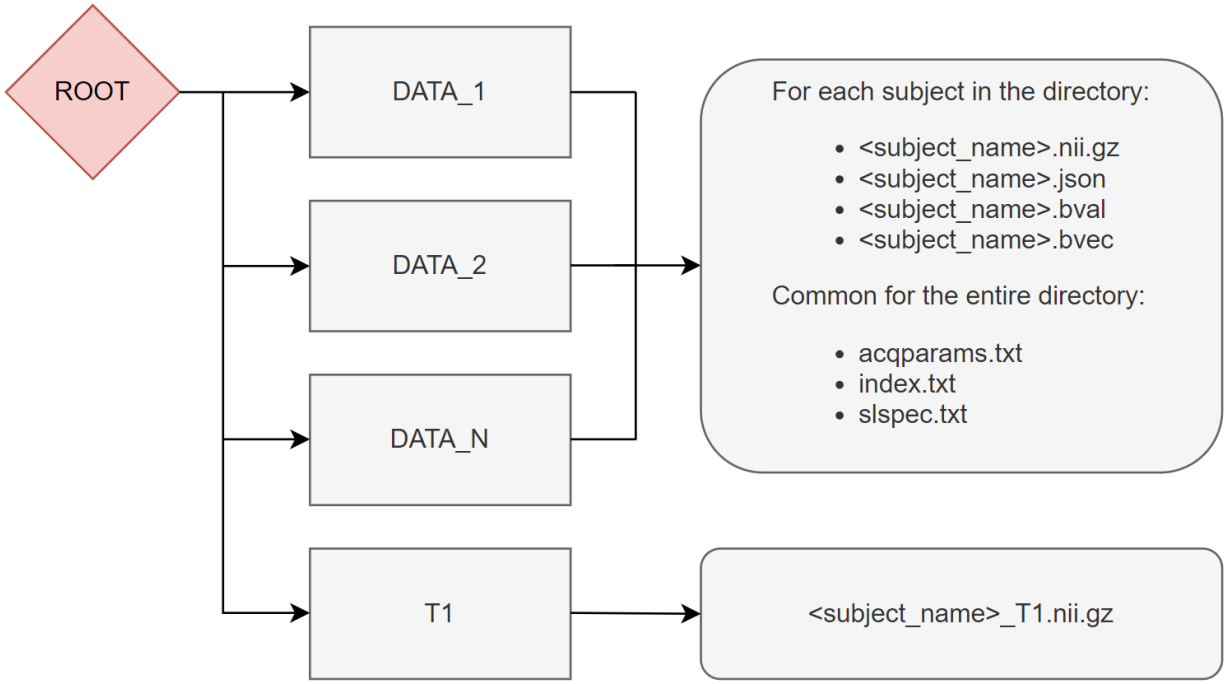
The first step to enable ElikoPy is to import it and initialise the ElikoPy object "*study* " specific to the current study. The only required argument for the constructor is the path to the root directory of the project.

```
import elikopy
import elikopy.utils

f_path="/PROJECTS/"
dic_path="/PROJECTS/static_files/mf_dic/fixed_rad_dist.mat"

study = elikopy.core.Elikopy(f_path)
study.patient_list()
```

The root directory must have the following structure during the initialisation



The T1 structural images as well as the acqparams, index and slspec files are optional. However, if they are not available, some processing steps might be not available (this is usually specified by a note in the documentation). These files can be generated as explained in the following links:

- acqparams.txt and index.txt : Eddy FSL acqp
- slspec.txt : Eddy FSL slspec

4.2 Preprocessing

The following code block show how to preproccess the dMRI data. By default only the brain extraction is enabled in the preprocessing but we recommend you to enable more preprocessing as described in the detailled guide (see *Preprocessing of diffusion images*).

```
8 study.preproc()
```

4.3 whitematter mask

The following code block computes a white matter mask for each subject from its T1 structural image (if available). If the T1 is not available, the mask is computed using the anisotropic power map generated from the diffusion data.

```
9 study.white_mask()
```

4.4 Microstructural metrics computation

The following code block computes microstructural metrics from the four microstructural model available in ElikoPy.

```
study.dti()
study.noddi()
study.diamond()
study.fingerprinting()
```

4.5 Statistical Analysis

In the following code block, fractional anisotropy (FA) from DTI along other additional metrics are registered into a common space. The registration is computed using the FA and the mathematical transformation is applied to other metrics.

Afterwards, the randomise_all function performs group wise statistic for the defined metrics along extraction of individual region wise value for each subject into csv files.

```
grp1=[1]
grp2=[2]




study.regall_FA(grp1=grp1,grp2=grp2)

additional_metrics={'_noddi_odi':'noddi','_mf_fvf_tot':'mf','_diamond_kappa':'diamond'}
study.regall(grp1=grp1,grp2=grp2, metrics_dic=additional_metrics)

metrics={'dti':'FA','_noddi_odi':'noddi','_mf_fvf_tot':'mf','_diamond_kappa':'diamond'}
study.randomise_all(metrics_dic=metrics)
```

4.6 Data Exportation

The export function is used to "revert" the folder structure, instead of using a subject specific folder tree, data are exported into a metric specific folder tree. In this example, only metrics computed from the dti model are exported.

```
study.export(raw=False, preprocessing=False, dti=True,
        noddi=False, diamond=False, mf=False, wm_mask=False, report=True)
```

Note: If you wish to learn more about the library and its validation, we recommend you to read the detailled guide and play around with the library.

4.7 Other parameters commonly available

The ElikoPy library has been made compatible with the slurm scheduler commonly present on HPC clusters. The use of the slurm scheduler can be controlled with the **slurm** parameters.

Associated options are:

- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – If not None, Topup will use additionnal parameters based on the supplied config file located at <topupConfig>. default=None Email adress to send notification if a task fails. default=None
- **slurm_timeout** - Replace the default slurm timeout used in the ElikoPy function by a custom timeout.
- **slurm_mem** - Replace the default amount of ram allocated to the slurm task by a custom amount of ram.
- **cpus** – Replace the default number of slurm cpus by a custom number of cpus.

The slurm option and slurm_email option can be globally define during the initialisation of the study object.

When processing a study, the processing for some subjects could fail for various reasons. The ElikoPy library provides two parameters destined to limit the amount of processing necessary to recover from these failures.

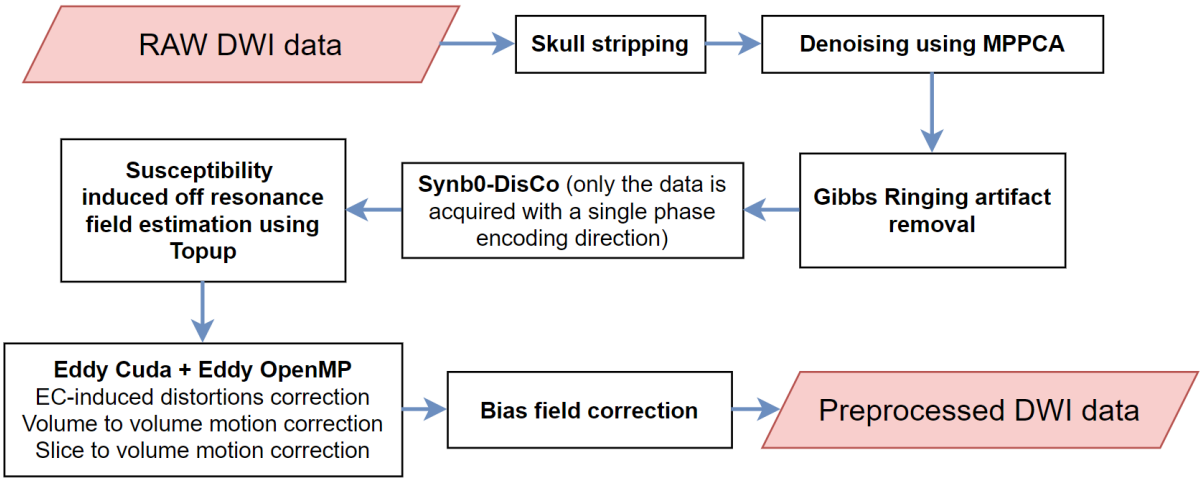
- **patient_list_m** – Define a subset of subjects to process instead of all the available subjects. example : ['patientID1','patientID2','patientID3']. default=None
- **starting_state** – Manually set which step of the function to start from. default=None

CHAPTER

FIVE

PREPROCESSING OF DIFFUSION IMAGES

The preprocessing stage aims at correcting the non idealities affecting the diffusion data before computing the diffusion metrics. With the exception of skull stripping, all processing steps are optional and can be applied at the user discretion.



To preproccess the dMRI data, the following line of code is used. However, this results in the default preprocessing that encompass only the skull striping step. To perform more advanced preprocessing, we need to dive into the arguments of the preproc function.

```
study.preproc()
```

The arguments of the preproc function are given in the API : LINK. In this page, only the main arguments are explained in order to grasp the key aspects of preprocessing using ElikoPy.

5.1 Reslice

5.1.1 Description

If the raw data is not in its 'native' resolution, a reslicing process might be required. Usually, the MRI scanner performs automatic interpolation on the data in order to beautify the data since clinicians usually have a preference for high resolution images. However, the intrinsic resolution is not augmented by this interpolation. While somewhat useful to clinicians, the interpolation is usually not desirable for research. Using it means more computation time and uncorrelated noise becoming correlated which reduces the performances of MPPCA denoising algorithms. Moreover, interpolation is not desirable when performing Gibbs ringing correction. Reslicing is therefore a way to mitigate the effect of mandatory interpolation during the acquisition.

5.1.2 Related parameters

The reslicing step during the preprocessing can be activated using the reslice argument.

- **reslice** - If true, data will be resliced with a new voxel resolution of 2*2*2. default=False
- **reslice_addSlice** - If true, an additional empty slice will be added to each volume (might be useful for motion correction if one slice is dropped during the acquisition and the user still wants to perform easily the slice-to-volume motion correction). default=False

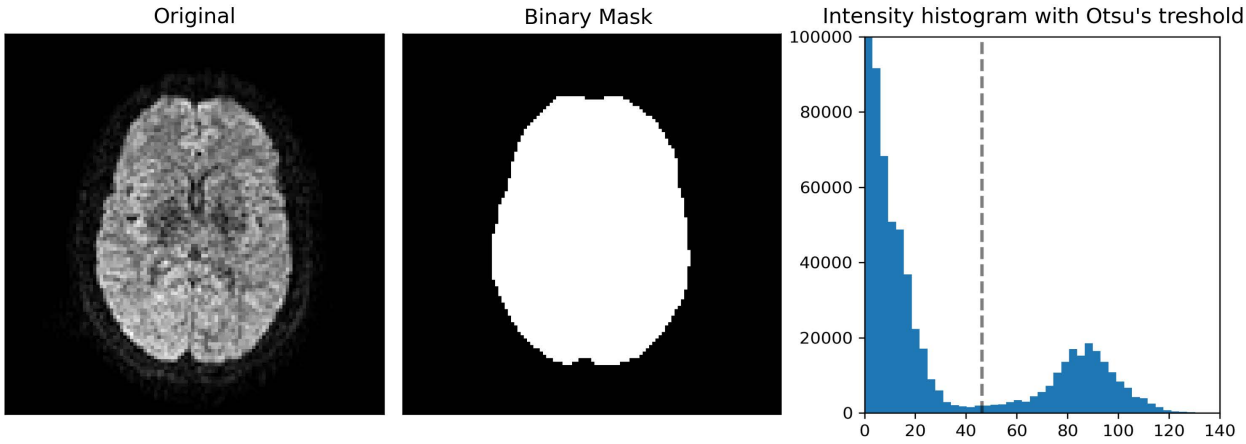
```
study.preproc(reslice=True,reslice_addSlice=False)
```

5.2 Brain Extraction

5.2.1 Description

The brain is extracted from the skull and other tissues surrounding the brain to increase the processing efficiency of subsequent steps and it is generally required before using other image processing algorithms. At the end of the preprocessing, a final brain mask readjusted in regard of all the applied preprocessing steps is also provided as output.

The mask is computed using median_otsu from DiPy.



5.2.2 Related parameters

The brain extraction is the only mandatory step and cannot be disabled. However, it is possible to change the parameters of the method

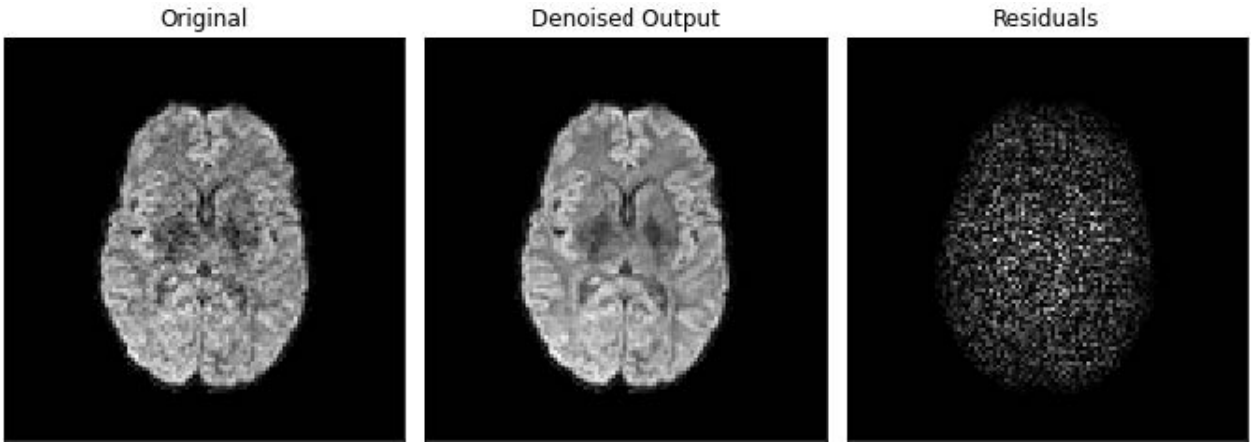
- **bet_median_radius** - Radius (in voxels) of the applied median filter during brain extraction. default=2
- **bet_numpass** - Number of pass of the median filter during brain extraction. default=1
- **bet_dilate** - Number of iterations for binary dilation during brain extraction. default=2

```
study.preproc(bet_median_radius=2, bet_numpass=2, bet_dilate=2)
```

5.3 MPPCA Denoising

5.3.1 Description

To reduce Rician noise typically found in MR images, the input images are denoised using the Marchenko-Pastur PCA technique as implemented in DiPy. Since the noise in diffusion data is spatially dependent in the case of multichannel receive coils, Principal component analysis of Marchenko-Pastur (MPPCA) noise-only distribution provides an accurate and fast method of noise evaluation and reduction. This methods has been chosen since it is a fast denoising algorithm that does not blur the image or create artifact.



5.3.2 Related parameters

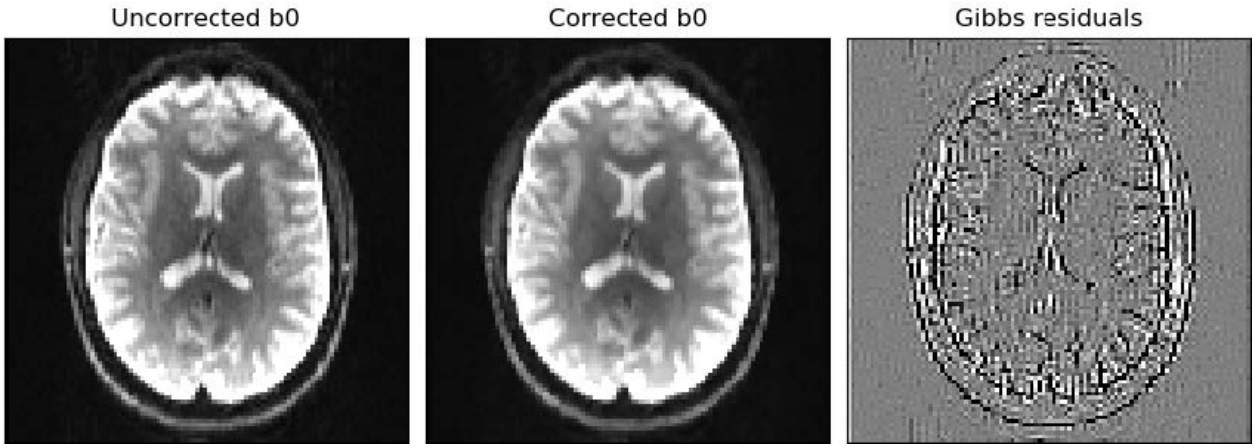
The denoising step during the preprocessing can be activated using the denoising argument.

```
study.preproc(denoising=True)
```

5.4 Gibbs Ringing Correction

5.4.1 Description

In general, in the context of diffusion-weighted imaging, derived diffusion-based estimates are affected by Gibbs oscillations. To correct for this, gibbs_removal from DiPy is used. This algorithm models the truncation of k-space as a convolution with a sinc-function in the image space. The severity of ringing artifacts thus depends on how the sampling of the sinc function occurs. The gibbs_removal function reinterpolate the image based on local, subvoxel-shifts to sample the ringing pattern at the zero-crossings of the oscillating sinc-function.



5.4.2 Related parameters

The Gibbs removal can be enabled using the gibbs argument.

```
study.preproc(gibbs=True)
```

Unless the data suffer heavily from Gibbs ringing artifacts, we do not advise to use the gibbs ringing removal step as it might blurr out small microstructural features.

5.5 Susceptibility field estimation

5.5.1 Description

Susceptibility distortions are created by differences in magnetic susceptibility near junctions of tissues. The susceptibility off resonance field is estimated using Topup from FSL. To do so, Topup needs data acquired with multiple phase encoding directions (at least 2). If only a single phase encoding direction is available, ElikoPy uses instead a generated synthetic volume based on a T1 structural image using Synb0-DisCo. This step only allows to **estimate** the susceptibility distortions, they are corrected at the same time as the eddy current distortions in the Eddy step below.

5.5.2 Related parameters

The susceptibility field estimation can be enabled using the topup argument.

- **topup** - true, Topup will estimate the susceptibility induced distortions. These distortions are corrected at the same time as EC-induced distortions if eddy=True. In the absence of images acquired with a reverse phase encoding direction, a T1 structural image is required. default=False
- **topupConfig** – If not None, Topup will use additionnal parameters based on the supplied config file located at <topupConfig>. default=None
- **forceSynb0DisCo** - If true, Topup will always estimate the susceptibility field using the T1 structural image. default=False

```
study.preproc(topup=True)
```

Note: If Topup is used, ElikoPy needs the acqparam and index files when generating the patient list : LINK (page getting started)

Note: If topup is enabled for data with a single phase encoding direction, a T1 structural image has to be provided when generating the patient list : LINK (page getting started)

5.6 Eddy and motion correction

5.6.1 Description

Motion, susceptibility and Eddy current induced distortions are artifacts with different origins but a similar effect i.e the displacement and deformation of the brain. They can therefore be jointly corrected. This is achieved using FSL Eddy. The susceptibility distortions are only corrected if they have been estimated during the topup step. By default only the inter-volume (volume-to-volume) motion is corrected but it is also possible to correct for intra-volume (slice-to-volume) motion.

5.6.2 Related parameters

The motion and distortion correction can be activated using the eddy argument. The number of iteration for the motion correction algorithm can also be changed.

```
study.preproc(eddy=True, niter=5)
```

In cases with large motion, inter-volume motion correction might not be sufficient and intra-volume correction is required. This option can be enabled using the s2v argument. The s2v input is a list of 4 parameters : [mporder,s2v_niter,s2v_lambda,s2v_interp]. The slice-to-volume motion correction is performed if mporder>0. These parameters are explained in depth in the FSL documentation (LINK). If N describes the number of excitations in a volume, setting mporder to N/4 while letting the other 3 parameters to their default values should provide good results in most situations. The slice-to-volume motion correction is only possible with cuda enabled.

Using the framework of Eddy FSL, it is also possible to replace outlier slices. This is done with the olrep argument which is a list of 4 parameters : [repol,ol_nstd,ol_nvox,ol_type]. The outlier replacement is performed if repol==True. These parameters are explained in depth in the FSL documentation.

```
study.preproc(eddy=True, niter=5, s2v=[6,5,1,'trilinear'], cuda=True, cuda_name='eddy_
↪cuda10.1', olrep=[True, 4, 250, 'sw'])
```

Note: If Eddy FSL is used, ElikoPy needs the acqparam and index files when generating the patient list : LINK (page getting started)

Note: If slice-to-volume motion correction is enabled, ElikoPy needs the slspec file when generating the patient list : LINK (page getting started)

5.7 Bias Field Correction

5.7.1 Description

Variability of the signal in tissues of the same type can affect microstructural metrics computation and brain segmentation algorithms. This can be corrected using the N4 Bias Field Correction algorithm.

5.7.2 Related parameters

The bias field correction can be activated using the biasfield argument. It is also possible to modify the parameters of the correction method.

- **biasfield_bsplineFitting** - Define the initial mesh resolution in mm and the bspline order of the biasfield correction tool.
- **biasfield_convergence** - Define the maximum number of iteration and the convergences threshold of the biasfield correction tool.

```
study.preproc(biasfield=True, biasfield_bsplineFitting=[100,3], biasfield_
↪convergence=[1000,0.001])
```

5.8 Report

By default, the preproc function outputs a quality report that contains quality control features for the processing. This can be disabled if needed.

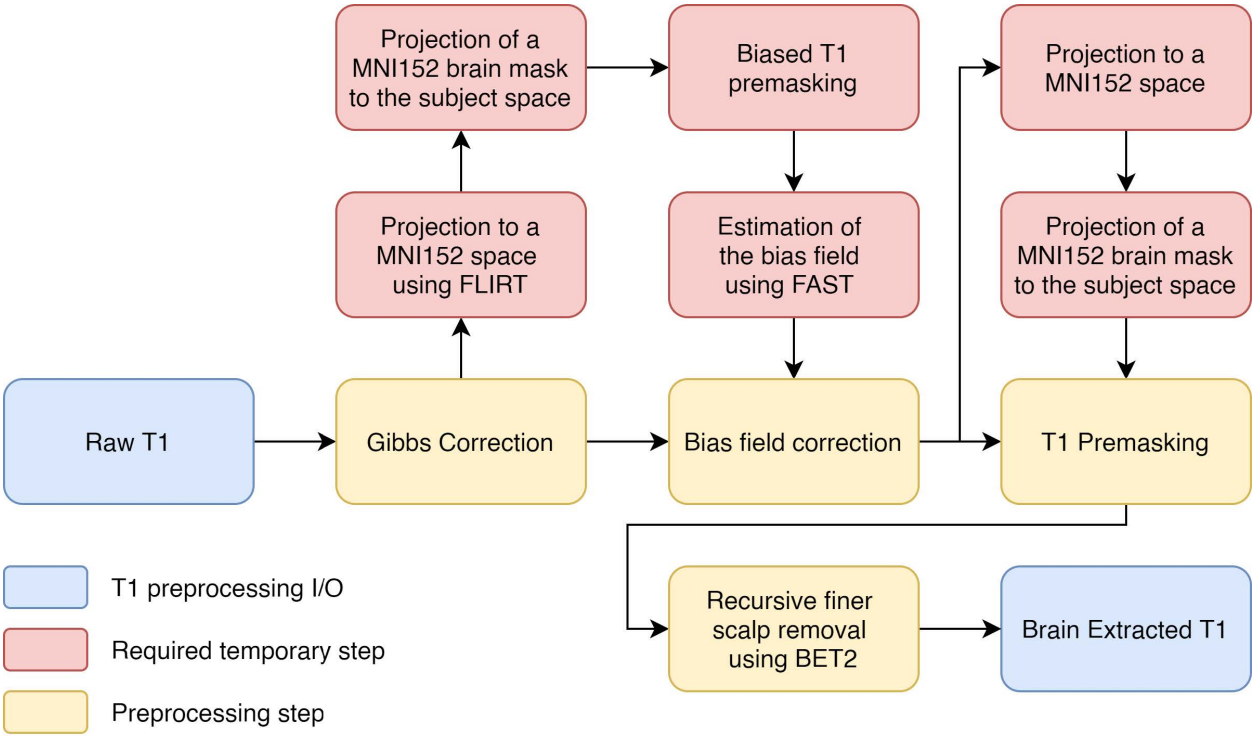
```
study.preproc(report=False)
```

CHAPTER

SIX

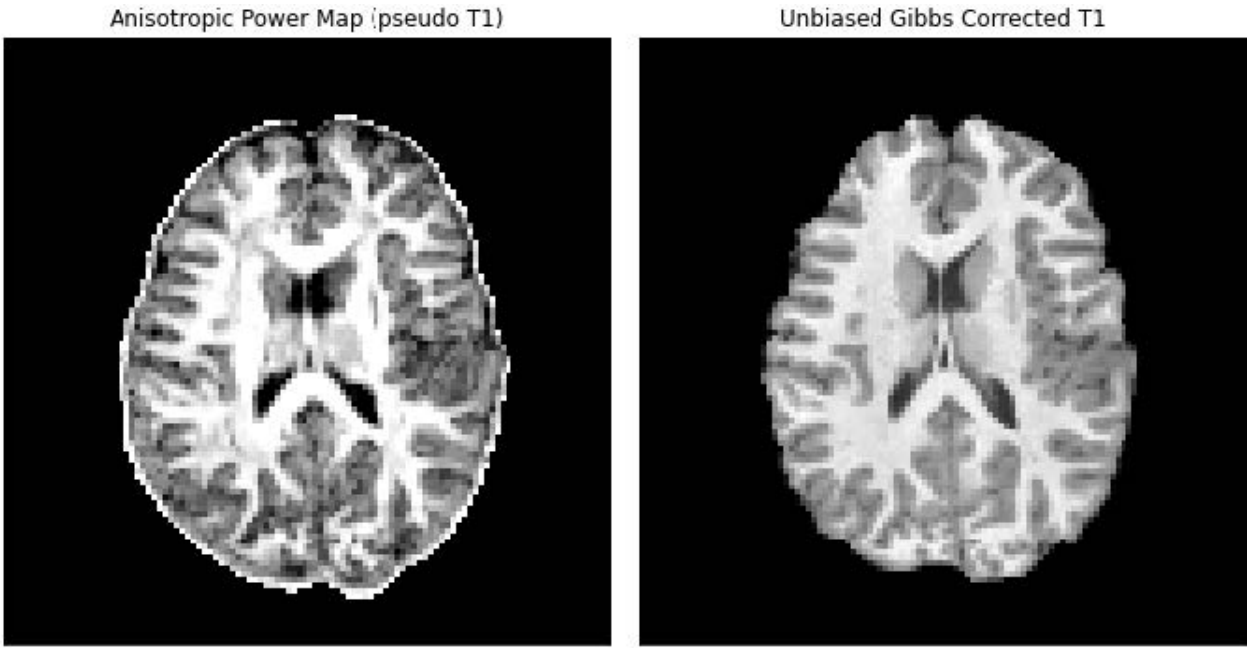
T1 PREPROCESSING

Providing a white matter mask is a useful step to accelerate microstructural features computation and more easily do tractography. The white_mask function of the ElikoPy library has been elaborated to perform this important step.



On the one hand, when a T1 image is available, a white matter mask can be computed from this data. Therefore, the T1 image is first preprocessed then segmented. Finally the segmented white matter mask is projected into the space of the preprocessed diffusion image.

On the other hand, when no T1 images are available, the white matter mask is directly computed from a segmentation of the diffusion data using Anisotropic Power (AP) map. In this case, no registrations are necessary.


Anisotropic Power Map (pseudo T1)
Unbiased Gibbs Corrected T1

CHAPTER

SEVEN

PROJECT AS AN EASY WAY TO MANAGE A STUDY

CHAPTER

EIGHT

PROJECT AS AN EASY WAY TO MANAGE A STUDY

CHAPTER

NINE

PROJECT AS AN EASY WAY TO MANAGE A STUDY

CHAPTER

TEN

EXAMPLES

CHAPTER

ELEVEN

CONTRIBUTING

ElikoPy is an open source project, meaning we have the challenge of limited resources. We are grateful for any support that you can offer. Helping other users, raising issues, helping write documentation, or contributing code are all ways to help!

11.1 Raise an Issue

For general bugs/issues, you can open an issue at the GitHub repo.

11.2 Write Documentation

We (like almost all open source software providers) have a documentation dilemma... We tend to focus on the code features and functionality before working on documentation. And there is very good reason for this: we want to share the love so nobody feels left out!

You can contribute to the documentation by raising an issue to suggest an improvement or by sending a pull request on our repository.

11.3 Contribute to the code

We use the traditional GitHub Flow to develop. This means that you fork the main repo, create a new branch to make changes, and submit a pull request (PR) to the master branch.

11.3.1 Step 1. Fork the repo

To contribute to ElikoPy, you should obtain a GitHub account and fork the ElikoPy repository. Once forked, clone your fork of the repo to your computer. (Obviously, you should replace `your-username` with your GitHub username.)

```
$ git clone https://github.com/your-username/elikopy.git && \
    cd elikopy/
```

11.3.2 Step 2. Checkout a new branch

Branches are a way of isolating your features from the main branch. Given that we've just cloned the repo, we will probably want to make a new branch from master in which to work on our new feature. Lets call that branch `new-feature`:

```
$ git checkout master && \
    git checkout -b new-feature
```

Note: You can always check which branch you are in by running `git branch`.

11.3.3 Step 3. Make your changes

On your new branch, go nuts! Make changes, test them, and when you are happy commit the changes to the branch:

```
$ git add file-changed1 file-changed2...

$ git commit -m "what changed?"
```

This commit message is important - it should describe exactly the changes that you have made. Good commit messages read like so:

```
$ git commit -m "changed function preproc in core.py to output new mask to fix #2"

$ git commit -m "updated docs about MF to close #10"
```

The tags `close #10` and `fix #2` are referencing issues that are posted on the upstream repo where you will direct your pull request. When your PR is merged into the master branch, these messages will automatically close the issues, and further, they will link your commits directly to the issues they intend to fix. This will help future maintainers understand your contribution, or (hopefully not) revert the code back to a previous version if necessary.

11.3.4 Step 4. Push your branch to your fork

When you are done with your commits, you should push your branch to your fork (and you can also continuously push commits here as you work):

```
$ git push origin new-feature
```

Note that you should always check the status of your branches to see what has been pushed (or not):

```
$ git status
```

11.3.5 Step 5. Submit a Pull Request

Once you have pushed your branch, then you can go to your fork (in the web GUI on GitHub) and submit a Pull Request. Regardless of the name of your branch, your PR should be submitted to the ElikoPy `master` branch. Submitting your PR will open a conversation thread for the maintainers of ElikoPy to discuss your contribution. At this time, the continuous integration that is linked with the code base will also be executed. If there is an issue, or if the maintainers suggest changes, you can continue to push commits to your branch and they will update the Pull Request.

11.3.6 Step 6. Keep your branch in sync

Cloning the repo will create an exact copy of the ElikoPy repository at that moment. As you work, your branch may become out of date as others merge changes into the upstream master. In the event that you need to update a branch, you will need to follow the next steps:

```
$ git remote add upstream https://github.com/Hyedryn/elikopy.git && # to add a new␣
↪remote named "upstream" \
    git checkout master && # or another branch to be updated \
    git pull upstream master && \
    git push origin master && # to update your fork \
    git checkout new-feature && \
    git merge master
```

CHAPTER

TWELVE

ELIKOPY PACKAGE

12.1 Submodules

12.2 elikopy.core module

Elikopy @author: qdessain, msimon

class elikopy.core.**Elikopy**(*folder_path*, *cuda=False*, *slurm=False*, *slurm_email='example@example.com'*)

Bases: object

Main class containing all the necessary function to process and preprocess a specific study.

diamond(*folder_path=None*, *patient_list_m=None*, *reportOnly=False*, *slurm=None*, *slurm_email=None*, *slurm_timeout=None*, *cpus=None*, *slurm_mem=None*)

Computes the DIAMOND metrics for each subject. The outputs are available in the directories <folder_path>/subjects/<subjects_ID>/dMRI/microstructure/diamond/.

example : study.diamond()

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **patient_list_m** – Define a subset of subjects to process instead of all the available subjects. example : ['patientID1','patientID2','patientID3']. default=None
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout of 14h by a custom timeout.
- **cpus** – Replace the default number of slurm cpus of 4 by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.
- **slurm_mem** – Replace the default amount of ram allocated to the slurm task (6096MO by cpu) by a custom amount of ram.

dti(*folder_path=None*, *patient_list_m=None*, *slurm=None*, *slurm_email=None*, *slurm_timeout=None*, *slurm_cpus=None*, *slurm_mem=None*)

Computes the DTI metrics for each subject using Weighted Least-Squares. The outputs are available in the directories <folder_path>/subjects/<subjects_ID>/dMRI/dti/.

example : study.dti()

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **patient_list_m** – Define a subset of subjects to process instead of all the available subjects. example : ['patientID1','patientID2','patientID3']. default=None
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout of 1h by a custom timeout.
- **slurm_cpus** – Replace the default number of slurm cpus of 1 by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.
- **slurm_mem** – Replace the default amount of ram allocated to the slurm task (8096MO by cpu) by a custom amount of ram.

export(*folder_path=None*, *raw=False*, *preprocessing=False*, *dti=False*, *noddi=False*, *diamond=False*, *mf=False*, *wm_mask=False*, *report=False*, *preprocessed_first_b0=False*, *patient_list_m=None*, *tractography=False*)

Allows to obtain in a single Export folder the outputs of specific processing steps for all subjects.

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **patient_list_m** – Define a subset of subjects to process instead of all the available subjects. example : ['patientID1','patientID2','patientID3']. default=None
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **raw** – If true, copy the raw data of each subject in the Export folder. default=FALSE
- **preprocessing** – If true, copy the preprocessed data of each subject in the Export folder. default=FALSE
- **dti** – If true, copy the DTI outputs of each subject in the Export folder. default=FALSE
- **noddi** – If true, copy the NODDI outputs of each subject in the Export folder. default=FALSE
- **diamond** – If true, copy the DIAMOND outputs of each subject in the Export folder. default=FALSE
- **mf** – If true, copy the MF outputs of each subject in the Export folder. default=FALSE
- **wm_mask** – If true, copy the white matter mask of each subject in the Export folder. default=FALSE
- **report** – If true, copy the quality control reports of each subject in the Export folder. default=FALSE
- **tractography** – If true, copy the tractography outputs of each subject in the Export folder. default=FALSE

fingerprinting(*dictionary_path=None*, *folder_path=None*, *CSD_bvalue=None*, *slurm=None*, *patient_list_m=None*, *slurm_email=None*, *slurm_timeout=None*, *cpus=None*, *slurm_mem=None*)

Computes the Microstructure Fingerprinting metrics for each subject. The outputs are available in the directories <folder_path>/subjects/<subjects_ID>/dMRI/microstructure/mf/.

example : study.fingerprinting(dictionary_path='my_dictionary')

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **dictionary_path** – Path to the dictionary of fingerprints (mandatory).
- **CSD_bvalue** – If the DIAMOND outputs are not available, the fascicles directions are estimated using a CSD with the images at the b-values specified in this argument. default=None
- **patient_list_m** – Define a subset of subjects to process instead of all the available subjects. example : ['patientID1','patientID2','patientID3']. default=None
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout of 20h by a custom timeout.
- **slurm_cpus** – Replace the default number of slurm cpus of 1 by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.
- **slurm_mem** – Replace the default amount of ram allocated to the slurm task (8096MO by cpu) by a custom amount of ram.

noddi(*folder_path=None*, *patient_list_m=None*, *force_brain_mask=False*, *slurm=None*, *slurm_email=None*, *slurm_timeout=None*, *cpus=None*, *slurm_mem=None*)

Computes the NODDI metrics for each subject. The outputs are available in the directories <folder_path>/subjects/<subjects_ID>/dMRI/microstructure/noddi/.

example : study.noddi()

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **patient_list_m** – Define a subset of subjects to process instead of all the available subjects. example : ['patientID1','patientID2','patientID3']. default=None
- **force_brain_mask** – Force the use of a brain mask even if a whitematter mask exist. default=False
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout of 10h by a custom timeout.
- **cpus** – Replace the default number of slurm cpus of 1 by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.
- **slurm_mem** – Replace the default amount of ram allocated to the slurm task (8096MO by cpu) by a custom amount of ram.

noddi_amico(*folder_path=None*, *patient_list_m=None*, *force_brain_mask=False*, *slurm=None*, *slurm_email=None*, *slurm_timeout=None*, *slurm_cpus=None*, *slurm_mem=None*)

Computes the NODDI amico metrics for each subject. The outputs are available in the directories <folder_path>/subjects/<subjects_ID>/dMRI/microstructure/noddi/.

example : study.noddi_amico()

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **patient_list_m** – Define a subset of subjects to process instead of all the available subjects. example : ['patientID1','patientID2','patientID3']. default=None
- **force_brain_mask** – Force the use of a brain mask even if a whitematter mask exist. default=False
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout of 10h by a custom timeout.
- **slurm_cpus** – Replace the default number of slurm cpus of 1 by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.
- **slurm_mem** – Replace the default amount of ram allocated to the slurm task (8096MO by cpu) by a custom amount of ram.

noddi_fix_icvf_thresholding(*folder_path=None*, *patient_list_m=None*, *fintra_threshold=0.99*, *fbundle_threshold=0.05*, *use_brain_mask=False*, *use_wm_mask=False*)

A function to quickly change the treshold value applied on the icvf metric of noddi without the needs of executing again the full noddi core function.

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **patient_list_m** – Define a subset of subjects to process instead of all the available subjects. example : ['patientID1','patientID2','patientID3']. default=None
- **fintra_threshold** – Threshold applied on the fintra. default=0.99
- **fbundle_threshold** – Threshold applied on the fbundle. default=0.05
- **use_brain_mask** – Set to 0 values outside the brain mask. default=False
- **use_wm_mask** – Set to 0 values outside the white matter mask. default=False

patient_list(*folder_path=None*, *bids_path=None*, *reverseEncoding=True*)

From the root folder containing data_1, data_2, … data_n folders with nifti files (and their corresponding bvals and bvecs), the Elikopy folder structure is created in a directory named 'subjects' inside folder_path. This step is mandatory. The validity of all the nifti present in the root folder is verified. If some nifti do not possess an associated bval and bvec file, they are discarded and the user is notified in a summary file named subj_error.json generated in the out sub-directory. All valid patients are stored in a file named patient_list.json. In addition to the nifti + bval + bvec, the data_n folders can also contain the json files (with the patient informations) as well as the acquparam, index and slspec files (used during the preprocessing). If these files are missing a warning is raised. In addition to the DW images, T1 structural images can be provided in a directory called 'T1' in the root folder.

example : study.patient_list()

Parameters

- **folder_path** – Path to the root folder of the study. default = study_folder
- **bids_path** – Path to the optional folder containing subjects' data in the BIDS format.
- **reverseEncoding** – Append reverse encoding direction to the DW-MRI data if available. default = True

patientlist_wrapper(*function*, *func_args*, *folder_path=None*, *patient_list_m=None*, *filename=None*, *function_name=None*, *slurm=False*, *slurm_email=None*, *slurm_timeout=None*, *cpus=None*, *slurm_mem=None*)

A wrapper function that apply a function given as an argument to every subject of the study. The wrapped function must takes two arguments as input, the patient_name and the path to the root of the study.

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **patient_list_m** – Define a subset of subjects to process instead of all the available subjects. example : ['patientID1','patientID2','patientID3']. default=None
- **function** – The pointer to the function (only without slurm /!)
- **func_args** – Additional arguments to pass to the wrapped function (only without slurm /!)
- **filename** – The name of the file containing the wrapped function (only with slurm /!)
- **function_name** – The name of the wrapped function (only with slurm /!)
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout of 20h by a custom timeout.
- **cpus** – Replace the default number of slurm cpus of 1 by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.
- **slurm_mem** – Replace the default amount of ram allocated to the slurm task (8096MO by cpu) by a custom amount of ram.

preproc(*folder_path=None*, *reslice=False*, *reslice_addSlice=False*, *denoising=False*, *gibbs=False*, *topup=False*, *topupConfig=None*, *forceSynb0DisCo=False*, *useGPUsynb0DisCo=False*, *eddy=False*, *biasfield=False*, *biasfield_bsplineFitting=[100, 3]*, *biasfield_convergence=[1000, 0.001]*, *patient_list_m=None*, *starting_state=None*, *bet_median_radius=2*, *bet_numpass=1*, *bet_dilate=2*, *cuda=None*, *cuda_name='eddy_cuda10.1'*, *s2v=[0, 5, 1, 'trilinear']*, *olrep=[False, 4, 250, 'sw']*, *slurm=None*, *slurm_email=None*, *slurm_timeout=None*, *cpus=None*, *slurm_mem=None*, *qc_reg=True*, *niter=5*, *slspec_gc_path=None*, *report=True*)

Performs data preprocessing. By default only the brain extraction is enabled. Optional preprocessing steps include : reslicing, denoising, gibbs ringing correction, susceptibility field estimation, EC-induced distortions and motion correction, bias field correction. The results are stored in the preprocessing subfolder of each study subject <folder_path>/subjects/<subjects_ID>/dMRI/preproc.

example : study.preproc(denoising=True, topup=True, eddy=True, biasfield=True)

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **reslice** – If true, data will be resliced with a new voxel resolution of 2*2*2. default=False
- **reslice_addSlice** – If true, an additional empty slice will be added to each volume (might be useful for motion correction if one slice is dropped during the acquisition and the user still wants to perform easily the slice-to-volume motion correction). default=False
- **denoising** – If true, MPPCA-denoising is performed on the data. default=False
- **gibbs** – If true, Gibbs ringing correction is performed. We do not advise to use this correction unless the data suffers from a lot of Gibbs ringing artifacts. default=False

- **topup** – If true, Topup will estimate the susceptibility induced distortions. These distortions are corrected at the same time as EC-induced distortions if eddy=True. In the absence of images acquired with a reverse phase encoding direction, a T1 structural image is required. default=False
- **topupConfig** – If not None, Topup will use additionnal parameters based on the supplied config file located at <topupConfig>. default=None
- **forceSynb0DisCo** – If true, Topup will always estimate the susceptibility field using the T1 structural image. default=False
- **eddy** – If true, Eddy corrects the EC-induced (+ susceptibility, if estimated) distortions and the motion. If these corrections are performed the acquparam and index files are required (see documentation). To perform the slice-to-volume motion correction the slspec file is also needed. default=False
- **biasfield** – If true, low frequency intensity non-uniformity present in MRI image data known as a bias or gain field will be corrected. default=False
- **biasfield_bsplineFitting** – Define the initial mesh resolution in mm and the bspline order of the biasfield correction tool. default=[100,3]
- **biasfield_convergence** – Define the maximum number of iteration and the convergences threshold of the biasfield correction tool. default=[1000,0.001]
- **patient_list_m** – Define a subset of subjects to process instead of all the available subjects. example : ['patientID1','patientID2','patientID3']. default=None
- **starting_state** – Manually set which step of the preprocessing to execute first. Could either be None, denoising, gibbs, topup, eddy, biasfield, report or post_report. default=None
- **bet_median_radius** – Radius (in voxels) of the applied median filter during brain extraction. default=2
- **bet_numpass** – Number of pass of the median filter during brain extraction. default=1
- **bet_dilate** – Number of iterations for binary dilation during brain extraction. default=2
- **cuda** – If true, eddy will run on cuda with the command name specified in cuda_name. default=False
- **cuda_name** – name of the eddy command to run when cuda==True. default="eddy_cuda10.1"
- **s2v** – list of parameters of Eddy for slice-to-volume motion correction (see Eddy FSL documentation): [mporder,s2v_niter,s2v_lambda,s2v_interp]. The slice-to-volume motion correction is performed if mporder>0, cuda is used and a slspec file is provided during the patient_list command. default=[0,5,1,'trilinear']
- **olrep** – list of parameters of Eddy for outlier replacement (see Eddy FSL documentation): [repol,ol_nstd,ol_nvox,ol_type]. The outlier replacement is performed if repol==True. default=[False, 4, 250, 'sw']
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout by a custom timeout.
- **cpus** – Replace the default number of slurm cpus by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.

- **slurm_mem** – Replace the default amount of ram allocated to the slurm task by a custom amount of ram.
- **qc_reg** – If true, the motion registration step of the quality control will be performed. We do not advise to use this argument as it increases the computation time. default=False
- **niter** – Define the number of iterations for eddy volume-to-volume. default=5
- **slspec_gc_path** – Path to the folder containing volume specific slice-specification for eddy. If not None, eddy motion correction with gradient cycling will be performed.
- **report** – If False, no quality report will be generated. default=True

randomise_all(*folder_path=None*, *randomise_numberofpermutation=5000*, *skeletonised=True*, *metrics_dic={'FA': 'dti', '_diamond_kappa': 'diamond', '_mf_fvf_tot': 'mf', '_noddi_odi': 'noddi'}*, *regionWiseMean=True*, *slurm=None*, *slurm_email=None*, *slurm_timeout=None*, *cpus=None*, *slurm_mem=None*)

Performs tract base spatial statistics (TBSS) between the data in grp1 and grp2 (groups are specified during the call to regall_FA) for each diffusion metric specified in the argument metrics_dic. The mean value of the diffusion metrics across atlases regions can also be reported in CSV files using the regionWiseMean flag. The used atlases are : the Harvard-Oxford cortical and subcortical structural atlases, the JHU DTI-based white-matter atlases and the MNI structural atlas It is mandatory to have performed regall_FA prior to randomise_all.

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **randomise_numberofpermutation** – Define the number of permutations. default=5000
- **skeletonised** – If True, randomize will be using only the white matter skeleton instead of the whole brain. default=True
- **metrics_dic** – Dictionnary containing the diffusion metrics to register in a common space. For each diffusion metric, the metric name is the key and the metric's folder is the value. default={'_noddi_odi':'noddi','_mf_fvf_tot':'mf','_diamond_kappa':'diamond'}
- **regionWiseMean** – If true, csv containing atlas-based region wise mean will be generated.
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout of 20h by a custom timeout.
- **cpus** – Replace the default number of slurm cpus of 1 by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.
- **slurm_mem** – Replace the default amount of ram allocated to the slurm task (8096MO by cpu) by a custom amount of ram.

regall(*folder_path=None*, *grp1=None*, *grp2=None*, *metrics_dic={'_diamond_kappa': 'diamond', '_mf_fvf_tot': 'mf', '_noddi_odi': 'noddi'}*, *slurm=None*, *slurm_email=None*, *slurm_timeout=None*, *cpus=None*, *slurm_mem=None*)

Register all the subjects diffusion metrics specified in the argument metrics_dic into a common space using the transformation computed for the FA with the regall_FA function. This is performed based on TBSS of FSL. It is mandatory to have performed regall_FA prior to regall.

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **grp1** – List of number corresponding to the type of the subjects to put in the first group.

- **grp2** – List of number corresponding to the type of the subjects to put in the second group.
- **metrics_dic** – Dictionnary containing the diffusion metrics to register in a common space. For each diffusion metric, the metric name is the key and the metric's folder is the value. default={'_noddi_odi':'noddi','_mf_fvf_tot':'mf','_diamond_kappa':'diamond'}
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout of 20h by a custom timeout.
- **cpus** – Replace the default number of slurm cpus of 1 by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.
- **slurm_mem** – Replace the default amount of ram allocated to the slurm task (8096MO by cpu) by a custom amount of ram.

regall_FA(*folder_path=None*, *grp1=None*, *grp2=None*, *starting_state=None*, *registration_type='-T'*, *postreg_type='-S'*, *prestats_treshold=0.2*, *slurm=None*, *slurm_email=None*, *slurm_timeout=None*, *cpus=None*, *slurm_mem=None*)

Register all the subjects Fractional Anisotropy into a common space, skeletonisedd and non skeletonised. This is performed based on TBSS of FSL. It is mandatory to have performed DTI prior to regall_FA.

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **grp1** – List of number corresponding to the type of the subjects to put in the first group.
- **grp2** – List of number corresponding to the type of the subjects to put in the second group.
- **starting_state** – Manually set which step of TBSS to execute first. Could either be None, reg, post_reg, prestats, design or randomise. default=None
- **registration_type** – Define the argument used by the tbss command tbss_2_reg. Could either by '-T', '-t' or '-n'. If '-T' is used, a FMRIB58_FA standard-space image is used. If '-t' is used, a custom image is used. If '-n' is used, every FA image is align to every other one, identify the "most representative" one, and use this as the target image.
- **postreg_type** – Define the argument used by the tbss command tbss_3_postreg. Could either by '-S' or '-T'. If you wish to use the FMRIB58_FA mean FA image and its derived skeleton, instead of the mean of your subjects in the study, use the '-T' option. Otherwise, use the '-S' option.
- **prestats_treshold** – Thresholds the mean FA skeleton image at the chosen threshold during prestats. default=0.2
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout of 20h by a custom timeout.
- **cpus** – Replace the default number of slurm cpus of 8 by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.
- **slurm_mem** – Replace the default amount of ram allocated to the slurm task (8096MO by cpu) by a custom amount of ram.

tbss(*folder_path=None*, *grp1=None*, *grp2=None*, *starting_state=None*, *last_state=None*, *registration_type='-T'*, *postreg_type='-S'*, *prestats_treshold=0.2*, *randomise_numberofpermutation=5000*, *slurm=None*, *slurm_email=None*, *slurm_timeout=None*, *slurm_tasks=None*, *slurm_mem=None*)

Performs tract base spatial statistics (TBSS) between the data in grp1 and grp2. The data type of each subject is specified by the subj_type.json file generated during the call to the patient_list function. The data type corresponds to the original directory of the subject (e.g. a subject that was originally in the folder data_2 is of type 2). It is mandatory to have performed DTI prior to tbss. This is function should not be used as it has been replaced by regall_FA, regall and randomise_all to allow for more flexibility.

example : study.tbss(grp1=[1,2], grp2=[3,4])

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **grp1** – List of number corresponding to the type of the subjects to put in the first group.
- **grp2** – List of number corresponding to the type of the subjects to put in the second group.
- **starting_state** – Manually set which step of TBSS to execute first. Could either be None, reg, post_reg, prestats, design or randomise. default=None
- **last_state** – Manually set which step of TBSS to execute last. Could either be None, preproc, reg, post_reg, prestats, design or randomise. default=None
- **registration_type** – Define the argument used by the tbss command tbss_2_reg. Could either by '-T', '-t' or '-n'. If '-T' is used, a FMRIB58_FA standard-space image is used. If '-t' is used, a custom image is used. If '-n' is used, every FA image is align to every other one, identify the "most representative" one, and use this as the target image.
- **postreg_type** – Define the argument used by the tbss command tbss_3_postreg. Could either by '-S' or '-T'. If you wish to use the FMRIB58_FA mean FA image and its derived skeleton, instead of the mean of your subjects in the study, use the '-T' option. Otherwise, use the '-S' option.
- **prestats_treshold** – Thresholds the mean FA skeleton image at the chosen threshold during prestats. default=0.2
- **randomise_numberofpermutation** – Define the number of permutations. default=5000
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout of 20h by a custom timeout.
- **slurm_tasks** – Replace the default number of slurm cpus of 8 by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.
- **slurm_mem** – Replace the default amount of ram allocated to the slurm task (8096MO by cpu) by a custom amount of ram.

white_mask(*folder_path=None*, *patient_list_m=None*, *corr_gibbs=True*, *forceUsePowerMap=False*, *debug=False*, *slurm=None*, *slurm_email=None*, *slurm_timeout=None*, *cpus=None*, *slurm_mem=None*)

Computes a white matter mask for each subject based on the T1 structural images or on the anisotropic power maps (obtained from the diffusion images) if the T1 images are not available. The outputs are available in the directories <folder_path>/subjects/<subjects_ID>/masks/. The T1 images can be gibbs ringing corrected.

example : study.white_mask()

Parameters

- **folder_path** – the path to the root directory. default=study_folder
- **patient_list_m** – Define a subset of subjects to process instead of all the available subjects. example : ['patientID1','patientID2','patientID3']. default=None
- **corr_gibbs** – If true, Gibbs ringing correction is performed on the T1 images. default=True
- **forceUsePowerMap** – Force the use of an AnisotropicPower map for the white matter mask generation. default=False
- **debug** – If true, additional intermediate output will be saved. default=False
- **slurm** – Whether to use the Slurm Workload Manager or not (for computer clusters). default=value_during_init
- **slurm_email** – Email adress to send notification if a task fails. default=None
- **slurm_timeout** – Replace the default slurm timeout of 3h by a custom timeout.
- **cpus** – Replace the default number of slurm cpus of 1 by a custom number of cpus of using slum, or for standard processing, its the number of core available for processing.
- **slurm_mem** – Replace the default amount of ram allocated to the slurm task (8096MO by cpu) by a custom amount of ram.

elikopy.core.**dicom_to_nifti**(*folder_path*)

Convert dicom data into compressed nifti. Converted dicoms are then moved to a sub-folder named original_data. The niftis are named patientID_ProtocolName_SequenceName.

Parameters **folder_path** – Path to root folder containing all the dicoms

12.3 elikopy.individual_subject_processing module

elikopy.individual_subject_processing.**diamond_solo**(*folder_path*, *p*, *core_count=4*, *reportOnly=False*)

Computes the DIAMOND metrics for a single subject. The outputs are available in the directories <folder_path>/subjects/<subjects_ID>/dMRI/microstructure/diamond/.

Parameters

- **folder_path** – the path to the root directory.
- **p** – The name of the patient.
- **core_count** – Number of allocated cpu cores. default=4

elikopy.individual_subject_processing.**dti_solo**(*folder_path*, *p*)

Computes the DTI metrics for a single subject. The outputs are available in the directories <folder_path>/subjects/<subjects_ID>/dMRI/dti/.

Parameters

- **folder_path** – the path to the root directory.
- **p** – The name of the patient.

elikopy.individual_subject_processing.**mf_solo**(*folder_path*, *p*, *dictionary_path*, *CSD_bvalue=None*, *core_count=1*)

Perform microstructure fingerprinting and store the data in the <folder_path>/subjects/<subjects_ID>/dMRI/microstructure/mf/.

Parameters

- **folder_path** – the path to the root directory.
- **p** – The name of the patient.
- **dictionary_path** – Path to the dictionary of fingerprints (mandatory).
- **CSD_bvalue** – If the DIAMOND outputs are not available, the fascicles directions are estimated using a CSD with the images at the b-values specified in this argument. default=None
- **core_count** – Define the number of available core. default=1

elikopy.individual_subject_processing.**noddi_amico_solo**(*folder_path*, *p*)

Perform noddi amico on a single subject and store the data in the <folder_path>/subjects/<subjects_ID>/dMRI/microstructure/noddi_amico/.

Parameters

- **folder_path** – the path to the root directory.
- **p** – The name of the patient.

elikopy.individual_subject_processing.**noddi_solo**(*folder_path*, *p*, *force_brain_mask=False*, *lambda_iso_diff=3e-09*, *lambda_par_diff=1.7e-09*, *use_amico=False*, *core_count=1*)

Computes the NODDI metrics for a single. The outputs are available in the directories <folder_path>/subjects/<subjects_ID>/dMRI/microstructure/noddi/.

Parameters

- **folder_path** – the path to the root directory.
- **p** – The name of the patient.
- **force_brain_mask** – Force the use of a brain mask even if a whitematter mask exist. default=False
- **lambda_iso_diff** – Define the noddi lambda_iso_diff parameters. default=3.e-9
- **lambda_par_diff** – Define the noddi lambda_par_diff parameters. default=1.7e-9
- **use_amico** – If true, use the amico optimizer. default=FALSE
- **core_count** – Number of allocated cpu cores. default=1

elikopy.individual_subject_processing.**preproc_solo**(*folder_path*, *p*, *reslice=False*, *reslice_addSlice=False*, *denoising=False*, *gibbs=False*, *topup=False*, *topupConfig=None*, *forceSynb0DisCo=False*, *useGPUsynb0DisCo=False*, *eddy=False*, *biasfield=False*, *biasfield_bsplineFitting=[100, 3]*, *biasfield_convergence=[1000, 0.001]*, *starting_state=None*, *bet_median_radius=2*, *bet_numpass=1*, *bet_dilate=2*, *cuda=False*, *cuda_name='eddy_cuda10.1'*, *s2v=[0, 5, 1, 'trilinear']*, *olrep=[False, 4, 250, 'sw']*, *qc_reg=True*, *core_count=1*, *niter=5*, *report=True*, *slspec_gc_path=None*)

Performs data preprocessing on a single subject. By default only the brain extraction is enabled. Optional preprocessing steps include : reslicing, denoising, gibbs ringing correction, susceptibility field estimation, EC-induced distortions and motion correction, bias field correction. The results are stored in the preprocessing subfolder of the study subject <folder_path>/subjects/<subjects_ID>/dMRI/preproc.

Parameters

- **folder_path** – the path to the root directory.
- **p** – The name of the patient.
- **reslice** – If true, data will be resliced with a new voxel resolution of 2*2*2. default=False
- **reslice_addSlice** – If true, an additional empty slice will be added to each volume (might be useful for motion correction if one slice is dropped during the acquisition and the user still wants to perform easily the slice-to-volume motion correction). default=False
- **denoising** – If true, MPPCA-denoising is performed on the data. default=False
- **gibbs** – If true, Gibbs ringing correction is performed. We do not advise to use this correction unless the data suffers from a lot of Gibbs ringing artifacts. default=False
- **topup** – If true, Topup will estimate the susceptibility induced distortions. These distortions are corrected at the same time as EC-induced distortions if eddy=True. In the absence of images acquired with a reverse phase encoding direction, a T1 structural image is required. default=False
- **topupConfig** – If not None, Topup will use additionnal parameters based on the supplied config file located at <topupConfig>. default=None
- **forceSynb0DisCo** – If true, Topup will always estimate the susceptibility field using the T1 structural image. default=False
- **useGPUsynb0DisCo** – If true, Topup will estimate the susceptibility field with the T1 structural image using cuda. default=FALSE
- **eddy** – If true, Eddy corrects the EC-induced (+ susceptibility, if estimated) distortions and the motion. If these corrections are performed the acquparam and index files are required (see documentation). To perform the slice-to-volume motion correction the slspec file is also needed. default=False
- **biasfield** – If true, low frequency intensity non-uniformity present in MRI image data known as a bias or gain field will be corrected. default=False
- **biasfield_bsplineFitting** – Define the initial mesh resolution in mm and the bspline order of the biasfield correction tool. default=[100,3]
- **biasfield_convergence** – Define the maximum number of iteration and the convergences threshold of the biasfield correction tool. default=[1000,0.001]
- **starting_state** – Manually set which step of the preprocessing to execute first. Could either be None, denoising, gibbs, topup, eddy, biasfield, report or post_report. default=None
- **bet_median_radius** – Radius (in voxels) of the applied median filter during brain extraction. default=2
- **bet_numpass** – Number of pass of the median filter during brain extraction. default=1
- **bet_dilate** – Number of iterations for binary dilation during brain extraction. default=2
- **cuda** – If true, eddy will run on cuda with the command name specified in cuda_name. default=False
- **cuda_name** – name of the eddy command to run when cuda==True. default="eddy_cuda10.1"
- **s2v** – list of parameters of Eddy for slice-to-volume motion correction (see Eddy FSL documentation): [mporder,s2v_niter,s2v_lambda,s2v_interp]. The slice-to-volume motion correction is performed if mporder>0, cuda is used and a slspec file is provided during the patient_list command. default=[0,5,1,'trilinear']

- **olrep** – list of parameters of Eddy for outlier replacement (see Eddy FSL documentation): [repol,ol_nstd,ol_nvox,ol_type]. The outlier replacement is performed if repol==True. default=[False, 4, 250, 'sw']
- **qc_reg** – If true, the motion registration step of the quality control will be performed. We do not advise to use this argument as it increases the computation time. default=False
- **niter** – Define the number of iterations for eddy volume-to-volume. default=5
- **slspec_gc_path** – Path to the folder containing volume specific slice-specification for eddy. If not None, eddy motion correction with gradient cycling will be performed.
- **report** – If False, no quality report will be generated. default=True
- **core_count** – Number of allocated cpu cores. default=1

elikopy.individual_subject_processing.**report_solo**(*folder_path*, *patient_path*, *slices=None*, *short=False*)

Legacy report function.

Parameters

- **folder_path** – path to the root directory.
- **patient_path** – Name of the subject.
- **slices** – Add additional slices cut to specific volumes
- **short** – Only output raw data, preprocessed data and FA data.

elikopy.individual_subject_processing.**white_mask_solo**(*folder_path*, *p*, *corr_gibbs=True*, *core_count=1*, *forceUsePowerMap=False*, *debug=False*)

Computes a white matter mask for a single subject based on the T1 structural image or on the anisotropic power map (obtained from the diffusion images) if the T1 image is not available. The outputs are available in the directories <folder_path>/subjects/<subjects_ID>/masks/. The T1 images can be gibbs ringing corrected.

Parameters

- **folder_path** – the path to the root directory.
- **p** – The name of the patient.
- **corr_gibbs** – If true, Gibbs ringing correction is performed on the T1 image. default=True
- **core_count** – Number of allocated cpu cores. default=1
- **forceUsePowerMap** – Force the use of an AnisotropicPower map for the white matter mask generation. default=False
- **debug** – If true, additional intermediate output will be saved. default=False

12.4 elikopy.utils module

elikopy.utils.**anonymise_nifti**(*rootdir*, *anonymize_json*, *rename*)

Anonymise all nifti present in rootdir by removing the PatientName and PatientBirthDate (only month and day) in the json and renaming the nifti file name to the PatientID.

Parameters

- **rootdir** – Folder containing all the nifti to anonimyse.

- **anonymize_json** – If true, edit the json to remove the PatientName and replace the PatientBirthDate by the year of birth.
- **rename** – If true, rename the nifti to the PatientID.

elikopy.utils.**export_files**(*folder_path*, *step*, *patient_list_m=None*)

Creates an export folder in the root folder containing the results of 'step' for each patient in a single folder

example : export_files('user/my_rootfolder', 'dMRI/microstructure/dti')

Parameters

- **folder_path** – root folder
- **step** – step to export
- **patient_list_m** – Define a subset a patient to process instead of all the available subjects.

elikopy.utils.**getJobsState**(*folder_path*, *job_list*, *step_name*)

Periodically checks the status of all jobs in the job_list. When a job status change to complete or a failing state. Write the status in the log and remove the job from the job_list. This function end when all jobs are completed or failed.

Parameters

- **folder_path** – The path to the root dir of the study (used to write the logs.txt file)
- **job_list** – The list of job to check for state update
- **step_name** – The string value of the prefix to put in the log file

elikopy.utils.**get_job_state**(*job_id*)

Retrieve the state of a job through the sacct bash command offered by the lurm Workload Manager. :param job_id: The id of the job to retrieve the state of. :return state: The string value representing the state of the job.

elikopy.utils.**get_patient_list_by_types**(*folder_path*, *type=None*)

Print the list of patient corresponding to a specfic type of patient.

Parameters

- **folder_path** – Path to the root folder of the study.
- **type** – The selected type

elikopy.utils.**inference**(*T1_path*, *b0_d_path*, *model*, *device*)

synb0DISCO adapted from https://github.com/MASILab/Synb0-DISCO

Parameters

- **T1_path** – Path to the normalized projected T1.
- **b0_d_path** – Path to the b0 atlases.
- **model** – DL Model
- **device** – Define if cuda or cpu is used.

elikopy.utils.**makedir**(*dir_path*, *log_path*, *log_prefix*)

Create a directory in the location specified by the dir_path and write the log in the log_path.

Parameters

- **dir_path** – The path to the directory to create.
- **log_path** – The path to the log file to write verbose data.
- **log_prefix** – The prefix to use in the log file.

elikopy.utils.**merge_all_reports**(*folder_path*)
Merge all subjects quality control reports into a single report.

Parameters **folder_path** – Path to the root folder of the study.

elikopy.utils.**merge_all_specific_reports**(*folder_path*, *merge_wm_report=False*, *merge_legacy_report=False*)
Merge all selected specific subject's report into a single big report.

Parameters

- **folder_path** – Path to the root folder of the study.
- **merge_wm_report** – Select wm report.
- **merge_legacy_report** – Select legacy report.

elikopy.utils.**randomise_all**(*folder_path*, *randomise_numberofpermutation=5000*, *skeletonised=True*, *metrics_dic={'FA': 'dti', '_diamond_kappa': 'diamond', '_mf_fvf_tot': 'mf', '_noddi_odi': 'noddi'}*, *core_count=1*, *regionWiseMean=True*)
Performs tract base spatial statistics (TBSS) between the data in grp1 and grp2 (groups are specified during the call to regall_FA) for each diffusion metric specified in the argument metrics_dic. The mean value of the diffusion metrics across atlases regions can also be reported in CSV files using the regionWiseMean flag. The used atlases are : the Harvard-Oxford cortical and subcortical structural atlases, the JHU DTI-based white-matter atlases and the MNI structural atlas It is mandatory to have performed regall_FA prior to randomise_all.

Parameters

- **folder_path** – path to the root directory.
- **randomise_numberofpermutation** – Define the number of permutations. default=5000
- **skeletonised** – If True, randomize will be using only the white matter skeleton instead of the whole brain. default=True
- **metrics_dic** – Dictionnary containing the diffusion metrics to register in a common space. For each diffusion metric, the metric name is the key and the metric's folder is the value. default={'_noddi_odi':'noddi','_mf_fvf_tot':'mf','_diamond_kappa':'diamond'}
- **regionWiseMean** – If true, csv containing atlas-based region wise mean will be generated.
- **core_count** – Number of allocated cpu core. default=1

elikopy.utils.**regall**(*folder_path*, *grp1*, *grp2*, *core_count=1*, *metrics_dic={'_diamond_kappa': 'diamond', '_mf_fvf_tot': 'mf', '_noddi_odi': 'noddi'}*)
Register all the subjects diffusion metrics specified in the argument metrics_dic into a common space using the transformation computed for the FA with the regall_FA function. This is performed based on TBSS of FSL. It is mandatory to have performed regall_FA prior to regall.

Parameters

- **folder_path** – path to the root directory.
- **grp1** – List of number corresponding to the type of the subjects to put in the first group.
- **grp2** – List of number corresponding to the type of the subjects to put in the second group.
- **metrics_dic** – Dictionnary containing the diffusion metrics to register in a common space. For each diffusion metric, the metric name is the key and the metric's folder is the value. default={'_noddi_odi':'noddi','_mf_fvf_tot':'mf','_diamond_kappa':'diamond'}
- **core_count** – Define the number of available core. default=1

elikopy.utils.**regall_FA**(*folder_path*, *grp1*, *grp2*, *starting_state=None*, *registration_type='-T'*, *postreg_type='-S'*, *prestats_treshold=0.2*, *core_count=1*)

Register all the subjects Fractional Anisotropy into a common space, skeletonisedd and non skeletonised. This is performed based on TBSS of FSL. It is mandatory to have performed DTI prior to regall_FA.

Parameters

- **folder_path** – path to the root directory.
- **grp1** – List of number corresponding to the type of the subjects to put in the first group.
- **grp2** – List of number corresponding to the type of the subjects to put in the second group.
- **starting_state** – Manually set which step of TBSS to execute first. Could either be None, reg, post_reg, prestats, design or randomise. default=None
- **registration_type** – Define the argument used by the tbss command tbss_2_reg. Could either by '-T', '-t' or '-n'. If '-T' is used, a FMRIB58_FA standard-space image is used. If '-t' is used, a custom image is used. If '-n' is used, every FA image is align to every other one, identify the "most representative" one, and use this as the target image.
- **postreg_type** – Define the argument used by the tbss command tbss_3_postreg. Could either by '-S' or '-T'. If you wish to use the FMRIB58_FA mean FA image and its derived skeleton, instead of the mean of your subjects in the study, use the '-T' option. Otherwise, use the '-S' option.
- **prestats_treshold** – Thresholds the mean FA skeleton image at the chosen threshold during prestats. default=0.2
- **core_count** – Define the number of available core. default=1

elikopy.utils.**submit_job**(*job_info*)

Submit a job to the Slurm Workload Manager using a crafted sbatch.

Parameters **job_info** – The parameters to use in the sbatch.

Return job_id The id of the submited job.

elikopy.utils.**synb0DisCo**(*folder_path*, *topuppath*, *patient_path*, *starting_step=None*, *topup=True*, *gpu=True*)

synb0DISCO adapted from https://github.com/MASILab/Synb0-DISCO

Parameters

- **folder_path** – path to the root directory.
- **topuppath** – Path to the subject's topup folder.
- **patient_path** – Name of the subject.
- **starting_step** – Define the starting step, usefull if previous step had already been run.
- **topup** – If true, topup will be perfomed after synb0Disco.
- **gpu** – If true, torch will use the gpu.

Return type object

elikopy.utils.**tbss_utils**(*folder_path*, *grp1*, *grp2*, *starting_state=None*, *last_state=None*, *registration_type='-T'*, *postreg_type='-S'*, *prestats_treshold=0.2*, *randomise_numberofpermutation=5000*)

[Legacy] Performs tract base spatial statistics (TBSS) between the data in grp1 and grp2. The data type of each subject is specified by the subj_type.json file generated during the call to the patient_list function. The data type corresponds to the original directory of the subject (e.g. a subject that was originally in the folder data_2 is of type 2). It is mandatory to have performed DTI prior to tbss.

Parameters

- **folder_path** – path to the root directory.
- **grp1** – List of number corresponding to the type of the subjects to put in the first group.
- **grp2** – List of number corresponding to the type of the subjects to put in the second group.
- **starting_state** – Manually set which step of TBSS to execute first. Could either be None, reg, post_reg, prestats, design or randomise. default=None
- **last_state** – Manually set which step of TBSS to execute last. Could either be None, preproc, reg, post_reg, prestats, design or randomise. default=None
- **registration_type** – Define the argument used by the tbss command tbss_2_reg. Could either by '-T', '-t' or '-n'. If '-T' is used, a FMRIB58_FA standard-space image is used. If '-t' is used, a custom image is used. If '-n' is used, every FA image is align to every other one, identify the "most representative" one, and use this as the target image.
- **postreg_type** – Define the argument used by the tbss command tbss_3_postreg. Could either by '-S' or '-T'. If you wish to use the FMRIB58_FA mean FA image and its derived skeleton, instead of the mean of your subjects in the study, use the '-T' option. Otherwise, use the '-S' option.
- **prestats_treshold** – Thresholds the mean FA skeleton image at the chosen threshold during prestats. default=0.2
- **randomise_numberofpermutation** – Define the number of permutations. default=5000

12.5 Module contents

CHAPTER

THIRTEEN

LICENSE

The project is licensed under the GNU AGPLv3 license:

```
                    GNU AFFERO GENERAL PUBLIC LICENSE
                       Version 3, 19 November 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <https://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The GNU Affero General Public License is a free, copyleft license for
software and other kinds of works, specifically designed to ensure
cooperation with the community in the case of network server software.

  The licenses for most software and other practical works are designed
to take away your freedom to share and change the works.  By contrast,
our General Public Licenses are intended to guarantee your freedom to
share and change all versions of a program--to make sure it remains free
software for all its users.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
them if you wish), that you receive source code or can get it if you
want it, that you can change the software or use pieces of it in new
free programs, and that you know you can do these things.

  Developers that use our General Public Licenses protect your rights
with two steps: (1) assert copyright on the software, and (2) offer
you this License which gives you legal permission to copy, distribute
and/or modify the software.

  A secondary benefit of defending all users' freedom is that
improvements made in alternate versions of the program, if they
receive widespread use, become available for other developers to
incorporate.  Many developers of free software are heartened and
encouraged by the resulting cooperation.  However, in the case of
software used on network servers, this result may fail to come about.
The GNU General Public License permits making a modified version and
```

(continues on next page)

(continued from previous page)

```
letting the public access it on a server without ever releasing its
source code to the public.

  The GNU Affero General Public License is designed specifically to
ensure that, in such cases, the modified source code becomes available
to the community.  It requires the operator of a network server to
provide the source code of the modified version running there to the
users of that server.  Therefore, public use of a modified version, on
a publicly accessible server, gives the public access to the source
code of the modified version.

  An older license, called the Affero General Public License and
published by Affero, was designed to accomplish similar goals.  This is
a different license, not a version of the Affero GPL, but Affero has
released a new version of the Affero GPL which permits relicensing under
this license.

  The precise terms and conditions for copying, distribution and
modification follow.

                       TERMS AND CONDITIONS

  0. Definitions.

  "This License" refers to version 3 of the GNU Affero General Public License.

  "Copyright" also means copyright-like laws that apply to other kinds of
works, such as semiconductor masks.

  "The Program" refers to any copyrightable work licensed under this
License.  Each licensee is addressed as "you".  "Licensees" and
"recipients" may be individuals or organizations.

  To "modify" a work means to copy from or adapt all or part of the work
in a fashion requiring copyright permission, other than the making of an
exact copy.  The resulting work is called a "modified version" of the
earlier work or a work "based on" the earlier work.

  A "covered work" means either the unmodified Program or a work based
on the Program.

  To "propagate" a work means to do anything with it that, without
permission, would make you directly or secondarily liable for
infringement under applicable copyright law, except executing it on a
computer or modifying a private copy.  Propagation includes copying,
distribution (with or without modification), making available to the
public, and in some countries other activities as well.

  To "convey" a work means any kind of propagation that enables other
parties to make or receive copies.  Mere interaction with a user through
a computer network, with no transfer of a copy, is not conveying.
```

(continues on next page)

(continued from previous page)

```
  An interactive user interface displays "Appropriate Legal Notices"
to the extent that it includes a convenient and prominently visible
feature that (1) displays an appropriate copyright notice, and (2)
tells the user that there is no warranty for the work (except to the
extent that warranties are provided), that licensees may convey the
work under this License, and how to view a copy of this License.  If
the interface presents a list of user commands or options, such as a
menu, a prominent item in the list meets this criterion.

  1. Source Code.

  The "source code" for a work means the preferred form of the work
for making modifications to it.  "Object code" means any non-source
form of a work.

  A "Standard Interface" means an interface that either is an official
standard defined by a recognized standards body, or, in the case of
interfaces specified for a particular programming language, one that
is widely used among developers working in that language.

  The "System Libraries" of an executable work include anything, other
than the work as a whole, that (a) is included in the normal form of
packaging a Major Component, but which is not part of that Major
Component, and (b) serves only to enable use of the work with that
Major Component, or to implement a Standard Interface for which an
implementation is available to the public in source code form.  A
"Major Component", in this context, means a major essential component
(kernel, window system, and so on) of the specific operating system
(if any) on which the executable work runs, or a compiler used to
produce the work, or an object code interpreter used to run it.

  The "Corresponding Source" for a work in object code form means all
the source code needed to generate, install, and (for an executable
work) run the object code and to modify the work, including scripts to
control those activities.  However, it does not include the work's
System Libraries, or general-purpose tools or generally available free
programs which are used unmodified in performing those activities but
which are not part of the work.  For example, Corresponding Source
includes interface definition files associated with source files for
the work, and the source code for shared libraries and dynamically
linked subprograms that the work is specifically designed to require,
such as by intimate data communication or control flow between those
subprograms and other parts of the work.

  The Corresponding Source need not include anything that users
can regenerate automatically from other parts of the Corresponding
Source.

  The Corresponding Source for a work in source code form is that
same work.

  2. Basic Permissions.
```

(continues on next page)

(continued from previous page)

```
  All rights granted under this License are granted for the term of
copyright on the Program, and are irrevocable provided the stated
conditions are met.  This License explicitly affirms your unlimited
permission to run the unmodified Program.  The output from running a
covered work is covered by this License only if the output, given its
content, constitutes a covered work.  This License acknowledges your
rights of fair use or other equivalent, as provided by copyright law.

  You may make, run and propagate covered works that you do not
convey, without conditions so long as your license otherwise remains
in force.  You may convey covered works to others for the sole purpose
of having them make modifications exclusively for you, or provide you
with facilities for running those works, provided that you comply with
the terms of this License in conveying all material for which you do
not control copyright.  Those thus making or running the covered works
for you must do so exclusively on your behalf, under your direction
and control, on terms that prohibit them from making any copies of
your copyrighted material outside their relationship with you.

  Conveying under any other circumstances is permitted solely under
the conditions stated below.  Sublicensing is not allowed; section 10
makes it unnecessary.

  3. Protecting Users' Legal Rights From Anti-Circumvention Law.

  No covered work shall be deemed part of an effective technological
measure under any applicable law fulfilling obligations under article
11 of the WIPO copyright treaty adopted on 20 December 1996, or
similar laws prohibiting or restricting circumvention of such
measures.

  When you convey a covered work, you waive any legal power to forbid
circumvention of technological measures to the extent such circumvention
is effected by exercising rights under this License with respect to
the covered work, and you disclaim any intention to limit operation or
modification of the work as a means of enforcing, against the work's
users, your or third parties' legal rights to forbid circumvention of
technological measures.

  4. Conveying Verbatim Copies.

  You may convey verbatim copies of the Program's source code as you
receive it, in any medium, provided that you conspicuously and
appropriately publish on each copy an appropriate copyright notice;
keep intact all notices stating that this License and any
non-permissive terms added in accord with section 7 apply to the code;
keep intact all notices of the absence of any warranty; and give all
recipients a copy of this License along with the Program.

  You may charge any price or no price for each copy that you convey,
and you may offer support or warranty protection for a fee.
```

(continues on next page)

(continued from previous page)

```
  5. Conveying Modified Source Versions.

  You may convey a work based on the Program, or the modifications to
produce it from the Program, in the form of source code under the
terms of section 4, provided that you also meet all of these conditions:

    a) The work must carry prominent notices stating that you modified
    it, and giving a relevant date.

    b) The work must carry prominent notices stating that it is
    released under this License and any conditions added under section
    7.  This requirement modifies the requirement in section 4 to
    "keep intact all notices".

    c) You must license the entire work, as a whole, under this
    License to anyone who comes into possession of a copy.  This
    License will therefore apply, along with any applicable section 7
    additional terms, to the whole of the work, and all its parts,
    regardless of how they are packaged.  This License gives no
    permission to license the work in any other way, but it does not
    invalidate such permission if you have separately received it.

    d) If the work has interactive user interfaces, each must display
    Appropriate Legal Notices; however, if the Program has interactive
    interfaces that do not display Appropriate Legal Notices, your
    work need not make them do so.

  A compilation of a covered work with other separate and independent
works, which are not by their nature extensions of the covered work,
and which are not combined with it such as to form a larger program,
in or on a volume of a storage or distribution medium, is called an
"aggregate" if the compilation and its resulting copyright are not
used to limit the access or legal rights of the compilation's users
beyond what the individual works permit.  Inclusion of a covered work
in an aggregate does not cause this License to apply to the other
parts of the aggregate.

  6. Conveying Non-Source Forms.

  You may convey a covered work in object code form under the terms
of sections 4 and 5, provided that you also convey the
machine-readable Corresponding Source under the terms of this License,
in one of these ways:

    a) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by the
    Corresponding Source fixed on a durable physical medium
    customarily used for software interchange.

    b) Convey the object code in, or embodied in, a physical product
    (including a physical distribution medium), accompanied by a
```

(continues on next page)

(continued from previous page)

```
    written offer, valid for at least three years and valid for as
    long as you offer spare parts or customer support for that product
    model, to give anyone who possesses the object code either (1) a
    copy of the Corresponding Source for all the software in the
    product that is covered by this License, on a durable physical
    medium customarily used for software interchange, for a price no
    more than your reasonable cost of physically performing this
    conveying of source, or (2) access to copy the
    Corresponding Source from a network server at no charge.

    c) Convey individual copies of the object code with a copy of the
    written offer to provide the Corresponding Source.  This
    alternative is allowed only occasionally and noncommercially, and
    only if you received the object code with such an offer, in accord
    with subsection 6b.

    d) Convey the object code by offering access from a designated
    place (gratis or for a charge), and offer equivalent access to the
    Corresponding Source in the same way through the same place at no
    further charge.  You need not require recipients to copy the
    Corresponding Source along with the object code.  If the place to
    copy the object code is a network server, the Corresponding Source
    may be on a different server (operated by you or a third party)
    that supports equivalent copying facilities, provided you maintain
    clear directions next to the object code saying where to find the
    Corresponding Source.  Regardless of what server hosts the
    Corresponding Source, you remain obligated to ensure that it is
    available for as long as needed to satisfy these requirements.

    e) Convey the object code using peer-to-peer transmission, provided
    you inform other peers where the object code and Corresponding
    Source of the work are being offered to the general public at no
    charge under subsection 6d.

  A separable portion of the object code, whose source code is excluded
from the Corresponding Source as a System Library, need not be
included in conveying the object code work.

  A "User Product" is either (1) a "consumer product", which means any
tangible personal property which is normally used for personal, family,
or household purposes, or (2) anything designed or sold for incorporation
into a dwelling.  In determining whether a product is a consumer product,
doubtful cases shall be resolved in favor of coverage.  For a particular
product received by a particular user, "normally used" refers to a
typical or common use of that class of product, regardless of the status
of the particular user or of the way in which the particular user
actually uses, or expects or is expected to use, the product.  A product
is a consumer product regardless of whether the product has substantial
commercial, industrial or non-consumer uses, unless such uses represent
the only significant mode of use of the product.

  "Installation Information" for a User Product means any methods,
```

(continues on next page)

(continued from previous page)

```
procedures, authorization keys, or other information required to install
and execute modified versions of a covered work in that User Product from
a modified version of its Corresponding Source.  The information must
suffice to ensure that the continued functioning of the modified object
code is in no case prevented or interfered with solely because
modification has been made.

  If you convey an object code work under this section in, or with, or
specifically for use in, a User Product, and the conveying occurs as
part of a transaction in which the right of possession and use of the
User Product is transferred to the recipient in perpetuity or for a
fixed term (regardless of how the transaction is characterized), the
Corresponding Source conveyed under this section must be accompanied
by the Installation Information.  But this requirement does not apply
if neither you nor any third party retains the ability to install
modified object code on the User Product (for example, the work has
been installed in ROM).

  The requirement to provide Installation Information does not include a
requirement to continue to provide support service, warranty, or updates
for a work that has been modified or installed by the recipient, or for
the User Product in which it has been modified or installed.  Access to a
network may be denied when the modification itself materially and
adversely affects the operation of the network or violates the rules and
protocols for communication across the network.

  Corresponding Source conveyed, and Installation Information provided,
in accord with this section must be in a format that is publicly
documented (and with an implementation available to the public in
source code form), and must require no special password or key for
unpacking, reading or copying.

  7. Additional Terms.

  "Additional permissions" are terms that supplement the terms of this
License by making exceptions from one or more of its conditions.
Additional permissions that are applicable to the entire Program shall
be treated as though they were included in this License, to the extent
that they are valid under applicable law.  If additional permissions
apply only to part of the Program, that part may be used separately
under those permissions, but the entire Program remains governed by
this License without regard to the additional permissions.

  When you convey a copy of a covered work, you may at your option
remove any additional permissions from that copy, or from any part of
it.  (Additional permissions may be written to require their own
removal in certain cases when you modify the work.)  You may place
additional permissions on material, added by you to a covered work,
for which you have or can give appropriate copyright permission.

  Notwithstanding any other provision of this License, for material you
add to a covered work, you may (if authorized by the copyright holders of
```

(continues on next page)

(continued from previous page)

```
that material) supplement the terms of this License with terms:

    a) Disclaiming warranty or limiting liability differently from the
    terms of sections 15 and 16 of this License; or

    b) Requiring preservation of specified reasonable legal notices or
    author attributions in that material or in the Appropriate Legal
    Notices displayed by works containing it; or

    c) Prohibiting misrepresentation of the origin of that material, or
    requiring that modified versions of such material be marked in
    reasonable ways as different from the original version; or

    d) Limiting the use for publicity purposes of names of licensors or
    authors of the material; or

    e) Declining to grant rights under trademark law for use of some
    trade names, trademarks, or service marks; or

    f) Requiring indemnification of licensors and authors of that
    material by anyone who conveys the material (or modified versions of
    it) with contractual assumptions of liability to the recipient, for
    any liability that these contractual assumptions directly impose on
    those licensors and authors.

  All other non-permissive additional terms are considered "further
restrictions" within the meaning of section 10.  If the Program as you
received it, or any part of it, contains a notice stating that it is
governed by this License along with a term that is a further
restriction, you may remove that term.  If a license document contains
a further restriction but permits relicensing or conveying under this
License, you may add to a covered work material governed by the terms
of that license document, provided that the further restriction does
not survive such relicensing or conveying.

  If you add terms to a covered work in accord with this section, you
must place, in the relevant source files, a statement of the
additional terms that apply to those files, or a notice indicating
where to find the applicable terms.

  Additional terms, permissive or non-permissive, may be stated in the
form of a separately written license, or stated as exceptions;
the above requirements apply either way.

  8. Termination.

  You may not propagate or modify a covered work except as expressly
provided under this License.  Any attempt otherwise to propagate or
modify it is void, and will automatically terminate your rights under
this License (including any patent licenses granted under the third
paragraph of section 11).
```

(continues on next page)

(continued from previous page)

```
  However, if you cease all violation of this License, then your
license from a particular copyright holder is reinstated (a)
provisionally, unless and until the copyright holder explicitly and
finally terminates your license, and (b) permanently, if the copyright
holder fails to notify you of the violation by some reasonable means
prior to 60 days after the cessation.

  Moreover, your license from a particular copyright holder is
reinstated permanently if the copyright holder notifies you of the
violation by some reasonable means, this is the first time you have
received notice of violation of this License (for any work) from that
copyright holder, and you cure the violation prior to 30 days after
your receipt of the notice.

  Termination of your rights under this section does not terminate the
licenses of parties who have received copies or rights from you under
this License.  If your rights have been terminated and not permanently
reinstated, you do not qualify to receive new licenses for the same
material under section 10.

  9. Acceptance Not Required for Having Copies.

  You are not required to accept this License in order to receive or
run a copy of the Program.  Ancillary propagation of a covered work
occurring solely as a consequence of using peer-to-peer transmission
to receive a copy likewise does not require acceptance.  However,
nothing other than this License grants you permission to propagate or
modify any covered work.  These actions infringe copyright if you do
not accept this License.  Therefore, by modifying or propagating a
covered work, you indicate your acceptance of this License to do so.

  10. Automatic Licensing of Downstream Recipients.

  Each time you convey a covered work, the recipient automatically
receives a license from the original licensors, to run, modify and
propagate that work, subject to this License.  You are not responsible
for enforcing compliance by third parties with this License.

  An "entity transaction" is a transaction transferring control of an
organization, or substantially all assets of one, or subdividing an
organization, or merging organizations.  If propagation of a covered
work results from an entity transaction, each party to that
transaction who receives a copy of the work also receives whatever
licenses to the work the party's predecessor in interest had or could
give under the previous paragraph, plus a right to possession of the
Corresponding Source of the work from the predecessor in interest, if
the predecessor has it or can get it with reasonable efforts.

  You may not impose any further restrictions on the exercise of the
rights granted or affirmed under this License.  For example, you may
not impose a license fee, royalty, or other charge for exercise of
rights granted under this License, and you may not initiate litigation
```

(continues on next page)

(continued from previous page)

```
(including a cross-claim or counterclaim in a lawsuit) alleging that
any patent claim is infringed by making, using, selling, offering for
sale, or importing the Program or any portion of it.

  11. Patents.

  A "contributor" is a copyright holder who authorizes use under this
License of the Program or a work on which the Program is based.  The
work thus licensed is called the contributor's "contributor version".

  A contributor's "essential patent claims" are all patent claims
owned or controlled by the contributor, whether already acquired or
hereafter acquired, that would be infringed by some manner, permitted
by this License, of making, using, or selling its contributor version,
but do not include claims that would be infringed only as a
consequence of further modification of the contributor version.  For
purposes of this definition, "control" includes the right to grant
patent sublicenses in a manner consistent with the requirements of
this License.

  Each contributor grants you a non-exclusive, worldwide, royalty-free
patent license under the contributor's essential patent claims, to
make, use, sell, offer for sale, import and otherwise run, modify and
propagate the contents of its contributor version.

  In the following three paragraphs, a "patent license" is any express
agreement or commitment, however denominated, not to enforce a patent
(such as an express permission to practice a patent or covenant not to
sue for patent infringement).  To "grant" such a patent license to a
party means to make such an agreement or commitment not to enforce a
patent against the party.

  If you convey a covered work, knowingly relying on a patent license,
and the Corresponding Source of the work is not available for anyone
to copy, free of charge and under the terms of this License, through a
publicly available network server or other readily accessible means,
then you must either (1) cause the Corresponding Source to be so
available, or (2) arrange to deprive yourself of the benefit of the
patent license for this particular work, or (3) arrange, in a manner
consistent with the requirements of this License, to extend the patent
license to downstream recipients.  "Knowingly relying" means you have
actual knowledge that, but for the patent license, your conveying the
covered work in a country, or your recipient's use of the covered work
in a country, would infringe one or more identifiable patents in that
country that you have reason to believe are valid.

  If, pursuant to or in connection with a single transaction or
arrangement, you convey, or propagate by procuring conveyance of, a
covered work, and grant a patent license to some of the parties
receiving the covered work authorizing them to use, propagate, modify
or convey a specific copy of the covered work, then the patent license
you grant is automatically extended to all recipients of the covered
```

(continues on next page)

(continued from previous page)

```
work and works based on it.

  A patent license is "discriminatory" if it does not include within
the scope of its coverage, prohibits the exercise of, or is
conditioned on the non-exercise of one or more of the rights that are
specifically granted under this License.  You may not convey a covered
work if you are a party to an arrangement with a third party that is
in the business of distributing software, under which you make payment
to the third party based on the extent of your activity of conveying
the work, and under which the third party grants, to any of the
parties who would receive the covered work from you, a discriminatory
patent license (a) in connection with copies of the covered work
conveyed by you (or copies made from those copies), or (b) primarily
for and in connection with specific products or compilations that
contain the covered work, unless you entered into that arrangement,
or that patent license was granted, prior to 28 March 2007.

  Nothing in this License shall be construed as excluding or limiting
any implied license or other defenses to infringement that may
otherwise be available to you under applicable patent law.

  12. No Surrender of Others' Freedom.

  If conditions are imposed on you (whether by court order, agreement or
otherwise) that contradict the conditions of this License, they do not
excuse you from the conditions of this License.  If you cannot convey a
covered work so as to satisfy simultaneously your obligations under this
License and any other pertinent obligations, then as a consequence you may
not convey it at all.  For example, if you agree to terms that obligate you
to collect a royalty for further conveying from those to whom you convey
the Program, the only way you could satisfy both those terms and this
License would be to refrain entirely from conveying the Program.

  13. Remote Network Interaction; Use with the GNU General Public License.

  Notwithstanding any other provision of this License, if you modify the
Program, your modified version must prominently offer all users
interacting with it remotely through a computer network (if your version
supports such interaction) an opportunity to receive the Corresponding
Source of your version by providing access to the Corresponding Source
from a network server at no charge, through some standard or customary
means of facilitating copying of software.  This Corresponding Source
shall include the Corresponding Source for any work covered by version 3
of the GNU General Public License that is incorporated pursuant to the
following paragraph.

  Notwithstanding any other provision of this License, you have
permission to link or combine any covered work with a work licensed
under version 3 of the GNU General Public License into a single
combined work, and to convey the resulting work.  The terms of this
License will continue to apply to the part which is the covered work,
but the work with which it is combined will remain governed by version
```

(continues on next page)

(continued from previous page)

```
3 of the GNU General Public License.

  14. Revised Versions of this License.

  The Free Software Foundation may publish revised and/or new versions of
the GNU Affero General Public License from time to time.  Such new versions
will be similar in spirit to the present version, but may differ in detail to
address new problems or concerns.

  Each version is given a distinguishing version number.  If the
Program specifies that a certain numbered version of the GNU Affero General
Public License "or any later version" applies to it, you have the
option of following the terms and conditions either of that numbered
version or of any later version published by the Free Software
Foundation.  If the Program does not specify a version number of the
GNU Affero General Public License, you may choose any version ever published
by the Free Software Foundation.

  If the Program specifies that a proxy can decide which future
versions of the GNU Affero General Public License can be used, that proxy's
public statement of acceptance of a version permanently authorizes you
to choose that version for the Program.

  Later license versions may give you additional or different
permissions.  However, no additional obligations are imposed on any
author or copyright holder as a result of your choosing to follow a
later version.

  15. Disclaimer of Warranty.

  THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY
APPLICABLE LAW.  EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT
HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY
OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
PURPOSE.  THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM
IS WITH YOU.  SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF
ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

  16. Limitation of Liability.

  IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS
THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY
GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE
USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF
DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD
PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS),
EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF
SUCH DAMAGES.

  17. Interpretation of Sections 15 and 16.
```

(continues on next page)

(continued from previous page)

```
  If the disclaimer of warranty and limitation of liability provided
above cannot be given local legal effect according to their terms,
reviewing courts shall apply local law that most closely approximates
an absolute waiver of all civil liability in connection with the
Program, unless a warranty or assumption of liability accompanies a
copy of the Program in return for a fee.

                     END OF TERMS AND CONDITIONS

            How to Apply These Terms to Your New Programs

  If you develop a new program, and you want it to be of the greatest
possible use to the public, the best way to achieve this is to make it
free software which everyone can redistribute and change under these terms.

  To do so, attach the following notices to the program.  It is safest
to attach them to the start of each source file to most effectively
state the exclusion of warranty; and each file should have at least
the "copyright" line and a pointer to where the full notice is found.

    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU Affero General Public License as published
    by the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU Affero General Public License for more details.

    You should have received a copy of the GNU Affero General Public License
    along with this program.  If not, see <https://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If your software can interact with users remotely through a computer
network, you should also make sure that it provides a way for users to
get its source.  For example, if your program is a web application, its
interface could display a "Source" link that leads users to an archive
of the code.  There are many ways you could offer source, and different
solutions will be better for different programs; see section 13 for the
specific requirements.

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU AGPL, see
<https://www.gnu.org/licenses/>.
```

PYTHON MODULE INDEX

e

elikopy, 53
elikopy.core, 37
elikopy.individual_subject_processing, 46
elikopy.utils, 49

INDEX

A

anonymise_nifti() (*in module elikopy.utils*), 49

D

diamond() (*elikopy.core.Elikopy method*), 37
diamond_solo() (*in module elikopy.individual_subject_processing*), 46
dicom_to_nifti() (*in module elikopy.core*), 46
dti() (*elikopy.core.Elikopy method*), 37
dti_solo() (*in module elikopy.individual_subject_processing*), 46

E

elikopy
    module, 53
Elikopy (*class in elikopy.core*), 37
elikopy.core
    module, 37
elikopy.individual_subject_processing
    module, 46
elikopy.utils
    module, 49
export() (*elikopy.core.Elikopy method*), 38
export_files() (*in module elikopy.utils*), 50

F

fingerprinting() (*elikopy.core.Elikopy method*), 38

G

get_job_state() (*in module elikopy.utils*), 50
get_patient_list_by_types() (*in module elikopy.utils*), 50
getJobsState() (*in module elikopy.utils*), 50

I

inference() (*in module elikopy.utils*), 50

M

makedir() (*in module elikopy.utils*), 50
merge_all_reports() (*in module elikopy.utils*), 50
merge_all_specific_reports() (*in module elikopy.utils*), 51
mf_solo() (*in module elikopy.individual_subject_processing*), 46
module
    elikopy, 53
    elikopy.core, 37
    elikopy.individual_subject_processing, 46
    elikopy.utils, 49

N

noddi() (*elikopy.core.Elikopy method*), 39
noddi_amico() (*elikopy.core.Elikopy method*), 39
noddi_amico_solo() (*in module elikopy.individual_subject_processing*), 47
noddi_fix_icvf_thresholding() (*elikopy.core.Elikopy method*), 40
noddi_solo() (*in module elikopy.individual_subject_processing*), 47

P

patient_list() (*elikopy.core.Elikopy method*), 40
patientlist_wrapper() (*elikopy.core.Elikopy method*), 40
preproc() (*elikopy.core.Elikopy method*), 41
preproc_solo() (*in module elikopy.individual_subject_processing*), 47

R

randomise_all() (*elikopy.core.Elikopy method*), 43
randomise_all() (*in module elikopy.utils*), 51
regall() (*elikopy.core.Elikopy method*), 43
regall() (*in module elikopy.utils*), 51
regall_FA() (*elikopy.core.Elikopy method*), 44
regall_FA() (*in module elikopy.utils*), 51
report_solo() (*in module elikopy.individual_subject_processing*), 49

S

submit_job() (*in module elikopy.utils*), 52
synb0DisCo() (*in module elikopy.utils*), 52

T

tbss() (*elikopy.core.Elikopy method*), 44
tbss_utils() (*in module elikopy.utils*), 52

W

white_mask() (*elikopy.core.Elikopy method*), 45
white_mask_solo() (*in module elikopy.individual_subject_processing*), 49