PyDesigner

Release 0.3

Siddhartha Dhiman, Joshua Teves, Kayti Keith

Apr 10, 2020

INSTALLATION

1 Welcome to the official PyDesigner project! 3

2 Notable Features 5

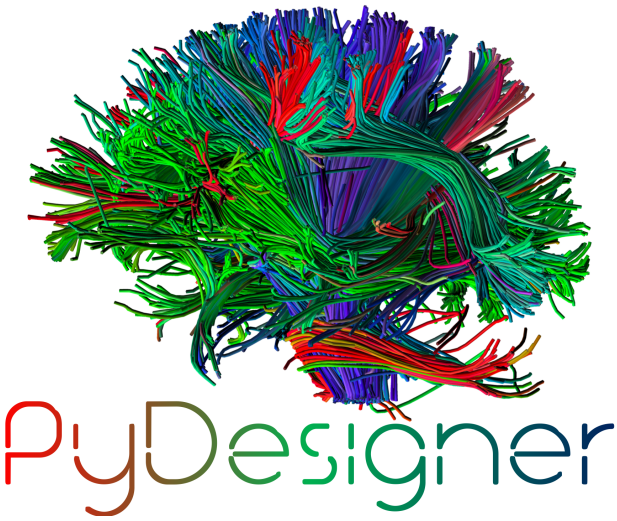
3 Indices and tables 27

CHAPTER

ONE

WELCOME TO THE OFFICIAL PYDESIGNER PROJECT!

PyDesigner was inspired by NYU's DESIGNER dMRI preprocessing pipeline to bring pre- and post- processing to every MRI imaging scientist. With PyDesigner, users are no longer confined to specific file types, operating systems, or complicated scripts just to extract DTI or DKI parameters – PyDesigner makes this easy, and you will love it!



CHAPTER

TWO

NOTABLE FEATURES

- **100% Python-based** scripts
- **Minimized package dependencies** for small package footprint
- Preprocessing designed to **boost SNR**
- **Accurate and fast** DTI and DKI metrics via cutting-edge algorithms
- **One-shot** preprocessing to parameter extraction
- **Cross-platform compatibility** between Windows, Mac and Linux using Docker
- Highly flexible and **easy to use**
- **Easy install** with *pip*
- Input **file-format agnostic** – works with .nii, .nii.gz, .mif and dicoms
- **Quality control metrics** to evaluate data integrity – SNR graphs and outlier voxels
- Uses the **latest techniques** from DTI/DKI literature

We welcome all DTI/DKI researchers to evaluate this software and pass on their feedback or issues through the Issues page of this project's GitHub repository. Additionally, you may join the M-AMA Slack channel for live support.

System Requirements Parallel processing in PyDesigner scales almost linearly with the nummber of CPU cores present. The application is also memory-intensive due to the number of parameter maps being computed.

Based on this evaluation, for processing a single DWI using PyDesigner, we recommend the following minimum system specifications:

- Ubuntu 18.04
- Intel i7-9700 or AMD Ryzen 1800X [8 cores]
- 16 GB RAM
- 12 GB free storage
- Nvidia CUDA-enabled GPU

2.1 Changelog

All notable changes to this project will be documented in this file or page

2.1.1 v0.32

Apr 10, 2020

Added:

- Intrinsic inter-axonal and mean extra-axonal diffusivity calculation to WMTI

Changed:

- Method `json2fslgrad` converted from class method to function definition
- Documentation update

Removed:

- None

2.1.2 v0.31

Apr 9, 2020

Added:

- NaN check in AWF calculculation that prevents further errors in intra-axonal and extra-axonal WMTI metrics computation

Changed:

- `designer.fitting.dwipy` input file detection method
- `Dockerfile_release` now deletes the correct temporary file to prevent build error

Removed:

- None

2.1.3 v0.3

Apr 8, 2020

Added:

- Head motion plot from on eddy_qc outputs
- Outlier plot from IRRLS outlier detection
- Updated documentation
- Option to reslice DWI with `--reslice [x,y,z]`

Changed:

- Flag `--epiboost [index]` changed to `--epi [n]`, where users can specify the number of reverse phase encoded B0 pairs to use in EPI correction. Non-indexed B0s were previously destructively removed from DWI, leading to incorrect weighing of B0s in tensor estimation. The new method now preserves all B0s, thereby allowing faster EPI distortion correction without degrading DTI/DKI maps.

- Documentation moved to ReadTheDocs
- Moved B0 production module from designer.preprocessing.brainmask to a separate function at `designer.preprocessing.extractmeanbzero()` that gets called by PyDesigner main. This allows a B0.nii to be produced regardless of the `--mask` flag.

Removed:

- Documentation inconsistencies

2.1.4 v0.2 [The Cupid Release]

Feb 26, 2020

Added:

- Installer for setup with `pip install .`
- Multiple file support: *.nii*, *.nii.gz*, *.dcm*, *.mif*
- reStructuredText styled documentation
- Ability to use `--resume` flag for DWI concatenation
- SNR plot to depict signal changes before and after preprocessing
- Full utilization of AVX instruction set on AMD machines
- WMTI parameters

Changed:

- Fixed topup series not being denoised

Removed:

- CSF masking; feature failed to work consistently

2.1.5 v0.11-dev

Dec 2, 2019

Added:

- None

Changed:

- Fixed bug in Dockerfile that prevented `pydesigner.py` from being found

Removed:

- None

2.1.6 v0.1-dev

Oct 22, 2019

Initial port of MATLAB code to Python. 200,000,000,000 BCE

2.2 PyDesigner Requirements

PyDesigner, currently, only requires the following three dependencies:

1. Python *3.6*, or above
2. FSL *6.0.2*, or above
3. MRtrix3, *3.0_RC3* or above

2.2.1 Linux and Mac Users

Unix-based system users are able to natively run all dependencies. Please proceed with the installation steps to configure PyD.

2.2.2 Windows Users

FSL and MRtrix3 are currently **not** *available on the Microsoft Windows* platform. Users running Windows are recommended to run the Docker image NeuroDock these interdependencies at near-native speed.

You may still proceed with the installation of PyDesigner Python modules to perform tensor fitting and map extraction.

2.3 FSL

FSL is a collection of tools and software used to process fMRI, MRI and DWI data. Visit their installation page for download and installation guide.

FSL 6.0.2 and above are recommended. All testing has been done with FSL 6.0.2. PyDesigner has not been tested with other versions of FSL.

To check your FSL version:

```
$ flirt -version
```

A return value of at least `FLIRT version 6.0` indicates successful installation of FSL, and that meets the PyD requirement.

2.4 MRtrix3 Installation

MRTRIX3 is another software suite aimed at analysis of DWI data. Here are some of their helpful pages.

1. Homepage
2. Download and Install

Confirm the success of installation with `mrinfo -version`. A valid output indicates successful installation.

2.5 Python

PyDesigner was built and tested on Python 3.7, so we enourage all users to adopt this version as well. While you may use the Python supplied by default on your OS, we highly enocurage users to adopt a Conda-based Python like Miniconda or Anaconda. Conda is a command line tool that allows the creation of separated environments with different python versions and packages. This of it as running multiple virtual machines on the a single host - you can easily switch between any for different needs, or run them simultaneously.

2.5.1 Download and Insall

Refer to either of these distributions' page for installation. This guide assumes a conda (Miniconda) installation for setting up Python. If you already have conda, or prefer using the default Python supplied by your OS, skip PyDesigner installation.

2.5.2 Update Conda

First, update conda with

```
$ conda update conda
```

2.5.3 Create new environment

Creating a conda environment is recommended as this will keep all of the dependencies required for this project isolated to just the conda environment called dmri. For more information about conda environments, see The Definitive Guide to Conda Environments. Next, create a conda environment specifically for dMRI preprocessing, called `dmri`. You can choose any name, but be sure to replace *dmri* in this guide with any name of your choice.

Next, execute the following two line to create a Python environment ready for PyD installation.

```
$ conda create -n dmri python=3.7
$ conda install -n dmri pip
```

The first line create an environment with Python v3.7, while the second line installs the PyPi package manager.

Once this is all set, you may proceed with the installation of PyD.

2.6 PyDesigner

PyD is an installable Python package deisgned to perform pre- and post- processing of dMRI acquisitions.

2.6.1 Download

You may clone the main PyDesigner repository for the latest build, or download the build version of your choice from the Releases tab.

To clone the PyDesigner repository, in terminal, run:

```
$ git clone https://github.com/m-ama/PyDesigner.git
```

2.6.2 Install

PyDesigner can be automatically installed with all dependencies by opening a CLI and changing directory to root PyDesigner directory, followed by

```
$ pip install .
```

Note: Remember to switch to your conda environement before parsing this command.

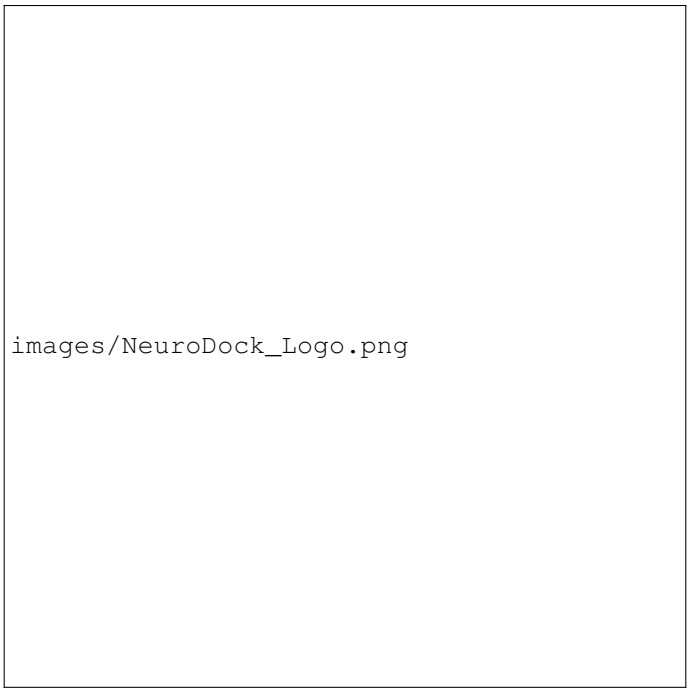
This will execute the `setup.py` script in root directory to automatically configure your Python environment for PyDesigner. When running the automated methods, PyDesigner can simply be called with the commad `pydesigner`.

Note: If you need a stable, tested and versioned build, download the most recent release from the Release tab. Click on Source code (zip) link and decompress (unzip) to any folder you desire.

2.7 NeuroDock

Docker is a contaner technology desgned to package an application and all its needs, such as libraried and other dependencies, into one package. We adapted PyDesigner and its dependencies for compatibility with the Docker Engine to bring DTI/DKI analyses to every one.

We bring you, NeuroDock



NeuroDock is a Docker image containing the most cutting-edge tools required for diffusion and kurtosis imaging. This container was designer for complete dMRI processing pipelines to be platform agnostic. NeuroDock was inspired by the lack of easily-accessible tools across various platforms. NeuroDock is 100% compatible across Windows, Linux, and Mac - while making available the full suite of FSL, MRtrix3 and PyDesigner commands.

2.7.1 Why Docker

By packaging fixed versions of FSL, MRtrix3, and PyDesigner, we are able to guarantee repeatbility and concistency across all platforms. Regardless of whether researchers are running Linux, Windows, or Mac OS, identical results can be replicated with Docker technology.

A side-effect to ensuring repeatiblity with Docker is that it becomes host operating system (OS) agnostic. This allows users to run FSL, MRtrix3, or PyDesigner commands at near-native speed, even on Microsoft Windows.

Additionally, researchers can easily deploy Docker containers to HPCs for rapid processing of large-cohort or longitudinal studies with ease.

2.7.2 Docker vs Virtual Machines

Okay, so you may ask, "why not just load up a VM?". You have a point. While the two technologies appear to be behaving the same way, at least on the surface level, their inner mechanisms are differ vastly.

Unlike a VM, rather than creating a whole virtual OS loaded with dependencies and other applications, Docker allows applications to share the same OS kernel, thereby providing a significant performance uplift while saving up storage space. With the removal of an entire guest OS in VMs, Docker containers save tons of computational resources that can be diverted towards better performance.

Now that you know some differences, it is time to move on to preparing the Docker image!

2.8 Docker Installation

Docker is relatively straightforward to install and run. Windows and Mac users are able to install Docker like any other GUI-based software package installtion. The installation is not dependent on console arguments, like Linux.

Please efer to the instructions below for links and guide.

2.8.1 Linux

Users may refer to the Docker Engine installation guide located here, for installation instructions on their Linux disribution. the steps covered below are targeted for Debian-based or Ubuntu distributions.

Uninstall Docker

1. Uninstall older version or any traces of existing Docker installations

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
```

Don't panic if `apt-get` returns an a warning about missing packages. It's good they are missing, since we're trying to purge existing installations of Docker

Install Docker Engine

Once all taces of existing Docker installation and dependencies have been purged, you may proceed with the following steps to install the Docker Engine - Community.

1. Update the debian package list with:

```
$ sudo apt-get Update
```

2. Install basic packages that enable installation of Docker Engine and its dependencies with:

```
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```

3. Add the Docker official GNU Privacy Guard (GPG) key to enable encryption and decryption of communication with the Docker server:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

You may verify this key by following the full guide on official Docker documentation, the link to which is located at the beginning of this page.

4. Add the stable Docker Engine repository to your package list with the command:

```
$ sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
```

Adding a repository to your Linux distribution allows the OS to pull software packages from the developers' servers. It directs the OS to the location where these packages are stored.

Then, update your package manager repository with the command:

```
$ sudo apt-get update
```

This updates the list of softwares your OS can fetch from various repositories.

5. Once your Debian-based system becomes aware of the Docker Engine, you may install it simply via the command:

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

6. Verify your Docker Engine installation with the command:

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

If the following information prints in the console window, your Docker Engine installation was sucessful.

```
Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
    1. The Docker client contacted the Docker daemon.
    2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
       (amd64)
    3. The Docker daemon created a new container from that image which runs the
       executable that produces the output you are currently reading.
    4. The Docker daemon streamed that output to the Docker client, which sent it
       to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
"$ docker run -it ubuntu bash"

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

You may now proceed with the fetching of NeuroDock Docker image.

2.8.2 Mac OS

1. Download Docker Docker Desktop for Mac.

2. Double-click on the downloaded *Docker.dmg* to start the install process. Follw all on-screen instrcutions and prompts.

3. Docker should start automatically, indicated by the whale icon in the status bar. Alternatively, you may verfiy whether Docker is running by parsing the following command in Terminal:

```
$ docker version
```

Or you may run the *hello-world* container to verify the installation:

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

If you information text being printed into the PowerShell windows, then Docker has been installed successfully.

2.8.3 Windows

1. Download Docker Desktop for Windows.

2. Double-click the *Docker for Windows Installer* to run the installer.

3. Docker should start automatically, indicated by the whale icon in the taskbar. Alternatively, you may verfiy whether Docker is running by parsing the following command in PowerShell.

```
$ docker version
```

Or you may run the *hello-world* container to verify the installation:

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

If you information text being printed into the PowerShell windows, then Docker has been installed successfully.

2.9 Docker Configuration

Docker can be configured in a wide-varietry of ways based on hardware resources available. Parameters such as CPU cores, RAM and storage can be assigned to Docker for running NeuroDock.

For validations purposes, the NeuroDock image was tested to work as intended on the following three systems:

Part	Machine A	Machine B	Machine C
Build	Apple iMac Pro	Custom	Custom
OS	Mac OS X Mojave	Ubuntu 18.04	Microsoft Windows 10 Pro
CPU	Intel Xeon W [8C/16T]	AMD Ryzen R9 2700X [8C/16T]	AMD Ryzen R9 2700X [8C/16T]
Memory	64 GB	16 GB	16 GB
Video	Raden Pro Vega 56 8 GB	Nvidia GTX 1080 8 GB	Nvidia GTX 1080 8 GB

We found identical results across the three operating systes on all these configurations.

2.9.1 Docker Preferences

Based on Docker's system requirements, we recommend assigning the following sysem resources to Docker:

Parameter	Value
CPUs	8
Memory	16.00 GB
Disk image size	32.00 GB

By default, Docker assigns itself half the number of available CPU cores and 2 GB of memory. Considering that the entire NeuroDock image is ~14.5 GB, we recommend at least double in disk image size. You may configure your Docker Engine to run on this configuration, or input your own values based on your processing needs. The following sections detail how to set these parameters.

Linux

CPU and memory access to Docker containers on Linux machines is manipulated via CFGS scheduler flags at run time. These flags are:

Flag	Description
`--cpus=<value>`	specify how many CPU cores to use
`-m` or `--memory`	specify the maximum amount of memory available to containers

For a more comprehensive list of manupulable system parameters for for Linux, please visit the Runtime options with Memory, CPUs, and GPUs page on Docker documentation.

Mac OS

Manipulating these three variables is very simple on Mac OS because these parameters are located in the GUI.

1. On the Docker icon in the status bar, right-click on the Docker icon, then **Preferences**.
2. Click on the **Resources** tab on the left

Fig. 1: Docker Mac preferences GUI; click on resources

3. The **Resourcs** menu will show you the configuration, please change them to desired valus. You may leave "Swap" at default.

Fig. 2: Docker Mac resources configuration

Windows

Similar to the Mac, the same sequence of steps apply for the Windows platform.

1. Right-click on the Dpcker icon in the taskbar, then click on **Preferences**.

2. Clock on the **Resources** tab on the left.

Fig. 3: Docker Windows preferences GUI; configure as desired

3. The **Resourcs** menu will show you the configuration, please change them to desired valus. You may leave "Swap" at default.

Setting the correct configuration will force Docker to not exceed these constraints. By splitting up CPU and memory loads, researchers can process multiple DWIs simultaneously.

2.9.2 GPU Support

At this time, there is no CUDA or ROCm GPU support. These feature are planned for a later release. Please use the non-Docker, native Linux configuration to utilize GPU for eddy and EPI correction.

2.10 Install NeuroDock

Aftet successfully installing and configuring docker, you can install the NeuroDock container in one of two ways:

1. Pulling pre-build image from Docker Hub with `docker pull [image]`
2. Building the image yourself with `docker build [path to image]`

The first option is the recommended method because prebuilt images are guaranteed to work and enhance repatibility even further. In additon, they are numbered version-controlled for referencing. Your copy of NeuroDock will be configured exactly the same other another person's.

The second option is intended for devopers who make frequent changes to the PyDesigner source code and wish to test their changes in a Docker environment. The Dockerfile script is designed to build a Docker image using PyDesigner in the root directory of the repository.

2.10.1 Docker Hub

Pulling pre-built NeuroDock is incredibly straight forward. Run the following command to pull NeuroDock.

```
$ docker pull docker pull dmri/neurodock:tagname
```

where `tagname` is the version you'd like to pull. To install NeuroDock v0.2, you would run the command

```
$ docker pull dmri/neurodock:v0.2
```

And that's it! All you have to do now is to wait for the NeuroDock image to finish downloading.

2.10.2 Local Build

Disclaimer It must be reiterated that this option is preserved for developoers; regular users are encoruaged to stay away from this method because there is no sematic versioning to referece.

1. Open up a command line interface and change directory to your PyDesigner repository

```
$ cd [PyDesginer Repo Path]
```

2. To build a Docker image using your local PyDesigner copy, run the command:

```
$ docker build -t [tagname] .
```

Here, `tagname` can be any name you wish to give this image. If you wish to build an image called neurodock, run the command:

```
$ docker build -t neurodock .
```

This will build a Docker image called NeuroDock based on your local Pyesigner repository.

2.11 Run NeuroDock

Congratulations, you've come this far. You've installed Docker and NeuroDock, and are probably wondering how what else to do...

You're done. Not even kidding! You can now start processing data with PyDesigner and NeuroDock. It's almost as if FSL, MRtrix3 and PyDesigner commands are built natively into your OS - be it Linux, Mac OS, or even Windows!

2.11.1 Intro to Docker Run

Use the following form of `docker run` command to call all command made availble by NeuroDock:

```
$ docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

where,

Flag	Definition
`[OPTIONS]`	docker options to use when running the container; common options are `-it`, `-v`, `-d`
`IMAGE`	image name to run; in this instance, this is `dmri/neurodock`
`[COMMAND]`	specify which NeuroDock commands to run; these can be FSL, MRtrix3 or PyDesigner commands
`[ARG]`	arguments for `[COMMAND]`

Users are encouraged to visit the Docker run reference documentation for more information on controlling the :code:`docker run ` command.

2.11.2 Practical Run

The section above convered a generic way to use the `docker run` command. For actual data analysis, we use the following options.

1. `-it --rm` to run docker in interative TTY mode. What this implies is that your NeuroDock command will run like any other OS commands such as `ipconfig`, `watch`, `ls` etc.

2. `-v` to mount the file system or folder to processing

Bind Mount

The second flag. `-v`, makes visible the host's local filesystem to a Docker container, which otherwise runs in a completely isolated system. By mounting a folder for NeuroDock, you are able to make it process data in said folder. The general guideline is to mount one subject folder at a time. It is advisable that users read through Docker's bind mounts to understand how Docker containers handle storage.

The correct syntax for the `-v` flag is:

```
-v [HOST PATH TO MOUNT]:[v]
```

Suppose a subject folder `bond_007` in need of processing is structured the following way:

```
bond_007
│
├── nifti
│ ├── bond_dwi.bval
│ ├── bond_dwi.bvec
│ ├── bond_dwi.bval
│ ├── bond_dwi.json
│ ├── bond_topup.json
│ └── bond_topup.nii
│
└── processed (empty dir)
```

This subject needs to be processed PyDesigner read the input nifti files in the `nifti` directory, and saves the outputs in the `processed` directory. Since both `nifti` and `processed` folders belong to a common parent directory, the `bond_007` directory can be mounted to give NeurDock access to both child directories simultaneously.

Here, the directory `bond_007` is the `[HOST PATH TO MOUNT]`, the directory that NeuroDock will not be able to see.

Next, we need to define where within the container this directory is mounted, `[TARGET AT WHICH TO MOUNT]`. You may simply mount this in the root NeuroDock directory at `/data`.

The flag to reflect this would then be:

```
-v /Users/sid/Desktop/bond_007:/data
```

This would make the contents of host directory `bond_007` available in the NeuroDock at `\data`. Say, for example, the nifti file `bond_dwi.nii`, is located in the host system at `/Users/sid/Desktop/bond_007/nifti/bond_dwi.nii`. If the above mounting scheme is used, the NeuroDock container will see this file in `/data/nifti/bond_dwi.nii`

This filesystem transformation is particularly important when writing scripts for automatic or batch processing of subject directories using the NeuroDock container.

Put it all together

Considering everything on this page, it becomes incredibly easy to process a subject using the NeuoDock container. Sticking to `bond_007` example above, and combining everthing so far, one could process Mr. Bond's DWI with the command:

```
$ docker run -it --rm -v /Users/sid/Desktop/bond_007:/data \
    dmri/neurodock pydesigner --standard \
    --output /data/processed \
    /data/nifti/bond_dwi.nii,/data/nifti/bond_topup.nii
```

This command runs the `--standard` PyDesigner pipeline on the input files `/Users/sid/Desktop/bond_007/nifti/bond_dwi.nii` and `/Users/sid/Desktop/bond_007/nifti/bond_topup.nii`, and saves all outputs into the directory `Users/sid/Desktop/bond_007/processed`

2.12 List of Flags

PyDesigner is extremely flexible when it comes to dMRI processing. Users can easily enable or disable various preprocessing steps without changing the overall sequence.

The list below covers all these flags.

2.12.1 IO Control

These flags allow control of the pipeline's I/O handling

-o DIR, --output DIR PyDesigner output directory

2.12.2 Preprocessing Control

Preprocessing contol flags allow users to tweak certain parts of the preprocessing pipeline, to accomodate all types of datasets.

-s, --standard	Runs the recommended preprocessing pipeline in order: denoise, degibbs, undistort, brain mask, smooth, rician
--denoise	Denoises input DWI
--extent	Shape of denoising extent matrix, defaults to 5,5,5
--reslice	Reslices input DWI and outputs to a specific resolution in mm or output dimensions
--interp	The interpolation method to use when resizing
--degibbs	Corrects Gibb's ringing
--undistort	Undistorts image using a suite of EPI distortion correction, eddy current correction, and co-registration. Does not run EPI correction if reverse phase encoding DWI is absent.

--rpe_pairs N	Speeds up topup if a reverse PE is present; specify the number (integer) of reverse PE direction B0 pairs to use
--mask	Computes a brain mask at 0.20 threshold by default
--maskthr	Specify FSL bet fractional intensity threshold for brain masking, defaults to 0.20
--user_mask	Provide path to user-generated brain mask in NifTi (.nii) format
--smooth	Smooths DWI data at a default FWHM of 1.25
--fwhm	Specify the FWHM at which to smooth, defaults to 1.25
--rician	Corrects Rician bias

2.12.3 Diffusion Tensor Control

Users may also tweak computations in estimating DTI or DKI parameters with the following flags.

--nofit	Performs preprocessing only, disables DTI/DKI parameter extraction
--noakc	Disables brute forced kurtosis tensor outlier rejection
--nooutliers	Disables IRLLS outlier detection
-w, --wmti	Enables IRLLS outlier detection, disable by default because calculations are experimental
--fit_constraints	Specify fitting constraints to use, defaults to 0,1,0
--noqc	Disables saving of quality control (QC) metrics
--median	Performs post processing median filter of final DTI/DKI maps. **WARNING: Use on a case-by-case basis for bad data only. When applied, the filter alters the values of most voxels, so it should be used with caution and avoided when data quality is otherwise adequate. While maps appear visually soother with this flag on, they may nonetheless be less accurate**

2.12.4 Pipeline Control

These are more general pipeline flags that interface directly with the user or machine.

--nthreads N	Specify number of CPU workers to use in processing, defaults to all physically available workers
--resume	Resumes preprocessing from an aborted or partial previous run
--force	Forces overwrite of existing output files
--verbose	Displays console output
--adv	Diables safety check to force run certain preprocessing steps **WARNING: This flag is for advanced users only who fully understand the MRI system and its outputs. Running with this flag could potentially yield inaccuracies in resulting DTI/DKI metrics**

2.13 List of Output Files

The number of output files generated by PyDesigner may seem very daunting at first. However, once a certain level of familiarity is achieved, it becomes very easy.

There are generally three types of outputs:

1. **Preocessing files:** files used in preprocessing; stored in root output directory 2. **Metric files:** DTI/DKI parameters maps, stored in `/metrics` folder 3. **QC Metrics:** files that enable data quality control; stored in `/metrics_qc`

The list of ever possible output file is given in the table below.

Filename	Description
Root Directory	
`B0.nii`	mean b0 image extracted from processed DWI (exists only if:code:*–mask* is
`brain_mask.nii`	brain mask extracted from B0.nii (exists only if `--mask` is used)
`dwi_preprocessed.nii`	fully preprocessed DWI NifTi file
`dwi_preprocessed.bval`	fully preprocessed DWI's BVAL file in FSL format
`dwi_preprocessed.bvec`	fully preprocessed DWI's BVEC file in FSL format
`dwi_preprocessed.json`	fully preprocessed DWI's BIDS sidecar
`dwi_raw.nii`	raw DWI NifTi file before preprocessing
`dwi_raw.bval`	raw DWI's BVAL file in FSL format
`dwi_raw.bvec`	raw DWI's BVEC file in FSL format
`dwi_raw.json`	raw DWI's BIDS sidecar
`noisemap.nii`	noisemap NifTi file produced from denoising (exists only if `--denoise` i
`working.mif`	MRtrix3 file formatted DWI that is being preprocessed
`log_command.json`	history of preprocessing steps and commands run of DWI
QC Metrics `root_dir/metrics_qc`	
`head_motion.png`	estimated head motion plotted from displacement field computed by EPI an
`outliers`	plot of percentage outliers detected by IRLLS outlier detection
`SNR.png`	snr plots of dwi_raw.nii and dwi_preprocessed.nii
`/fitting/outliers_akc.nii`	outliers detected by brute forced kurtosis tensor outlier rejection algorithm
`/fitting/outliers_irlls.nii`	outliers voxels detected by irlls outlier detection (4d nifti)
`/eddy`	all outputs of the eddy correction (exists if `--undistort` is used)
DTI/DKI Metrics `root_dir/metrics`	
`ad.nii`	axial diffusivity map (3d nifti)
`rd.nii`	radial diffusivity map (3d nifti)
`md.nii`	mean diffusivity map (3d nifti)
`fa.nii`	fractional anisotropy map (3d nifti)
`fe.nii`	first eigenvalues; represents the principal direction of water (4d nifti)
`trace.nii`	sum of diagnols of in diffusion tensor (3d nifti); the mean diffusivity (MD)
`ak.nii`	axial kurtosis map (3 nifti)
`rk.nii`	radial kurtosis map (3d nifti)
`mk.nii`	mean kurtosis map (3d nifti)
`kfa.nii`	kurtosis fractional anisotropy map (3d nifti)
`mkt.nii`	mean kurtosis tensor (3d nifti); alternative calculation for mean kurtosis
`DT.nii`	diffusion tensor (4d nifti; 6 three-dimensional volumes)
`KT.nii`	kurtosis tensor (4d nifti; 15 three dimensional volumes)
WMTI Metrics `root_dir/metrics`	
`wmti_awf.nii`	axonal water fraction (3d nifti)
`wmti_eas_ad`	extra-axonal axial diffusivity (3d nifti)
`wmti_eas_rd`	extra-axonal radial diffusivity (3d nifti)

Table 1 – continued from previous page

wmti_eas_md	extra-axonal mean diffusivity (3d nifti)
wmti_eas_tort	extra-axonal tortuosity (3d nifti)
wmti_ias_ad	intra-axonal axial diffusivity (3d nifti)
wmti_ias_rd	intra-axonal radial diffusivity (3d nifti)
wmti_ias_da	intra-axonal intrinsic diffusivity (3d nifti)
wmti_ias_tort	intra-axonal tortuosity (3d nifti)

All other files in folder /intermediate_nifti are used by PyDesigner for preprocessing flow control, especially to allow --resume or --force flags to work as intended.

2.14 Grants

The PyDesigner project is a result of effort from several contributors and support from grants. This section acknowledges all contributions to the project.

Table 2: List of supporting grants

Sponsor	Grant Number	Title
NIH	1R01AG05[illegible]150	Quantitative Neuroimaging Assessment of White Matter Integrity in the Context of Aging and AD
NIH	P20GM109[illegible]446	Institutional Award (IDeA) Center of Biomedical Research Excellence
NIA	1R01AG05[illegible]602	Assessing Brain Microstructure in Alzheimer's Disease with Advanced Diffusion MRI
NIDCD	R01DC014[illegible]24	Brain Connectivity Supporting Language Recovery in Aphasia
NINDS	R01NS110[illegible]47	Predicting Epilepsy Surgery Outcomes Using Neural Network Architecture

2.15 Contributors

Fig. 4: Siddhartha Dhiman, MSc
Research Specialist

The Center for Biomedical Imaging
Department of Neuroscience
Medical University of South Carolina

Fig. 5: Joshua Teves, BSc
Systems Programmer

The Center for Biomedical Imaging
Department of Neuroscience
Medical University of South Carolina

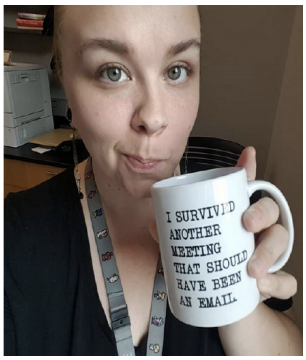


Fig. 6: Kayti Keith, BSc
Research Specialist

The Center for Biomedical Imaging
Department of Neuroscience
Medical University of South Carolina

Fig. 7: Benjamin Ades-Aron, MSc
PhD Student

Center for Biomedical Imaging
Department of Radiology
NYU School of Medicine
New York University

Fig. 8: Jens Jensen, PhD
Professor

The Center for Biomedical Imaging
Department of Neuroscience
Medical University of South Carolina

Fig. 9: Els Fieremans, PhD
Assistant Professor

Center for Biomedical Imaging
Department of Radiology
NYU School of Medicine
New York University

Fig. 10: Jelle Veraart, PhD
Assistant Professor

Center for Biomedical Imaging
Department of Radiology
NYU School of Medicine
New York University

Fig. 11: Vitria Adisetiyo, PhD
Staff Scientist

The Center for Biomedical Imaging
Department of Neuroscience
Medical University of South Carolina

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search