Articles Describing Code

Tractography analysis with the scilpy toolbox

Emmanuelle Renauld, MSc[1][a], Arnaud Boré, MSc[1], Charles Poirier, MSc[1], Alex Valcourt-Caron, MSc[1], Philippe Karan, MSc[1], Antoine Théberge, MSc[1], Guillaume Théaud, Ph.D[2], Manon Edde, Ph.D[1], Philippe Poulin, Ph.D[1], Gabriel Girard, Ph.D[1], Jean-Christophe Houde, MSc[3], Anthony Gagnon, MSc[1,4], Etienne St-Onge, Ph.D[5], Graham Little, Ph.D[1], Jon Haitz Legarreta, Ph.D[6,7], Stanislas Thoumyre, MSc[1,8], Gabrielle Grenier, MSc[1], Zineb El Yamani[1], Mario Ocampo Pineda, Ph.D[9], Matteo Battocchio, Ph.D[1,10], Vincent Beaudoin[1], Alexandre Joanisse[1], Laurent Petit, Ph.D[11,12], François Rheault, Ph.D[1,12]‡, Maxime Descoteaux, Ph.D[1,3,12]‡

[1] Computer Sciences Department, Université de Sherbrooke, [2] Neuroscience Research Axis, University of Montreal Hospital Research Center (CRCHUM), [3] Imeka Solutions Inc., Sherbrooke, Canada, [4] Pediatric Department, Université de Sherbrooke, [5] Computer Science and Engineering Department, Université du Québec en Outaouais, [6] Department of Radiology, Brigham and Women's Hospital, [7] Harvard Medical School, [8] Neurodegeneratives Diseases Institute, UMR 5293, Team 5 - CEA - CNRS, Université de Bordeaux, [9] Department of Biomedical Engineering, University of Basel, [10] Department of Computer Science, University of Verona, [11] CNRS, CEA, IMN, GIN, UMR 5293, Université de Bordeaux, [12] IRP OpTeam, CNRS Biologie, France and Université de Sherbrooke, Canada

Keywords: image processing, DWI, dMRI, DTI, fODF, tractography, connectivity, medical imaging, collaboration, open source

https://doi.org/10.52294/001c.154022

Aperture Neuro

Vol. 6, 2026

We present scilpy, an open-source Python library for diffusion magnetic resonance imaging (dMRI) and tractography. It provides a large collection of well-documented, user-friendly scripts that support nearly every stage of the dMRI processing pipeline, from preprocessing (such as denoising, registration, and local fiber reconstruction) to tractography and post-processing of tractograms, including connectivity and bundle analyses. Scilpy leverages the strength of DIPY, is well-tested through modern continuous integration practices, and is actively maintained.

INTRODUCTION

We present scilpy, an open-source Python library for diffusion magnetic resonance imaging (dMRI) and tractography processing.

DMRI allows analyzing water diffusion in biological tissues, offering valuable insight into their underlying microstructure. This information can be used to infer complex features, from local, voxel-wise properties to large-scale connectivity patterns, such as the organization of fibers in the brain white matter (WM). The earliest uses of dMRI included basic tractography through diffusion tensor imaging (DTI)[1] and the reconstruction of apparent diffusion coefficient (ADC) and fractional anisotropy (FA) maps.[1] In recent years, the neuroimaging community has developed advanced local reconstruction methods and tractography algorithms,[2,3] along with powerful tools for analyzing the resulting tractograms, which consist of sets of hundreds of thousands or even millions of streamlines. These include, among others, methods for bundle characterization, which involves analyzing the shape of a bundle or its underlying microstructural properties in a chosen population[4]; analysis of tractogram-related metrics (tractometry, profilometry), which focuses on how selected metrics evolve along different sections of a bundle or the whole-bundle[5]; and connectomics, which examines the strength of structural connections between regions of interest (ROIs).[6] The evolving landscape of analysis methods requires scientific developers to reimplement and provide continuous support to enable new discoveries (see Supplementary Table A1 for definitions of technical terms relevant to this work). In addition to the usual, well documented initial preprocessing steps, including denoising, registration, segmentation, or reconstruction of local anisotropy information, new analysis techniques include performing task-specific tractography, filtering out outlier streamlines, segmenting bundles, preparing connectivity matrices, computing bundle characteristics or underlying dMRI metrics, and more. Here, we present scilpy, a Python library designed to facilitate a wide range of processing steps, with a particular focus on post-tractography analysis. Scripts are well documented and user friendly: most options have suggested default values based on expert knowledge to help even newcomers use

‡ Joint last authors

a Corresponding Author:
Emmanuelle Renauld
emmanuelle.renauld@usherbrooke.ca

the scripts and obtain good results. Scilpy is updated regularly, allowing it to respond effectively to the new methods that are constantly being introduced in the computational neuroimaging domain, and allowing researchers to follow the most recent trends in analysis. It does not aim to replace existing libraries but rather to complement them. It finds its place alongside other well-known libraries and applications in dMRI (DIPY,[7] MRtrix3,[8] Tortoise,[9] DSI Studio,[10] Phybers[11]) and more broadly in MRI, including FreeSurfer,[12] AFNI,[13] FSL,[14] ANTs[15] (see Table 1). Scilpy supports many data formats (e.g., nifti, trk, tck), which makes it easy to use in conjunction with other software.

A general overview of scilpy is presented in section 1. The strength and possibilities from its numerous scripts are presented in section 2. In section 3, we showcase pipelines resolving several dMRI tasks of interest through a series of scilpy scripts. Section 4 details the organization and rigorous testing implemented in scilpy. Section 5 illustrates how scilpy can be used in many research collaborations. Finally, the value of scilpy for educational purposes is highlighted in section 6 and future perspectives are discussed in section 7.

SECTION 1: THE SCILPY LIBRARY

Scilpy has evolved from a private in-house repository into a renowned, robust, and publicly available library. It was created in 2012 as an internal collection of scripts of the Sherbrooke Connectivity Imaging Lab (SCIL). It was initially hosted on Bitbucket, then transferred to a public GitHub repository (https://github.com/scilus/scilpy) in February 2018, and the Bitbucket version was archived. It is supported by a team of over 30 contributors and developers, many of whom are students and collaborators of the SCIL, in Sherbrooke, Canada. Scilpy works on both Linux and macOS and can be installed through PyPI. Scilpy is coded in Python, an easy-to-read, open-source programming language. Some functions requiring faster processing are coded in Cython.

Scilpy aims to be easy to use, and thus both the scripts and the Python functions are well documented, and all descriptions can be found online through the automatically generated API documentation (https://scilpy.readthedocs.io). The website also provides a list of tutorials and examples of typical sequences in the use of the scripts. Scripts are prepared to offer a single task per script, making it easy to separate the scripts into categories (see Table 1), giving a clear overview of the possibilities. Scilpy provides a detailed explanation of required and optional arguments in all scripts. Default, well-tested values that work for a large set of healthy human adult brain analyses are provided. These values have been modified to analyse other populations, such as pediatric[16] or Alzheimer's[17] cohorts, but the scripts provide only limited support for animal brains. Users can access any script's documentation via its "–help" option in the command line. Additionally, the script scil_search_keywords helps look for keywords throughout all scripts. As for the Python functions, their docstrings include a summary, expected input parameters, and a list of outputs.

SECTION 2: RELEVANCE OF SCILPY FOR THE COMPUTATIONAL NEUROIMAGING COMMUNITY

Scilpy offers to the neuroimaging community a library with tools covering nearly all preprocessing or postprocessing steps in dMRI, from local modelling to bundle analysis. Users may wonder when to choose scilpy over other libraries. Scilpy does not aim to replace other available tools, but rather to complement them. A general overview of our scripts and how they compare with other libraries in the dMRI field is presented in Table 1.

The first necessary distinction is between scilpy and DIPY.[7] Scilpy serves as a platform for rapid development, offering access to new tools and enhancements that may not yet be integrated into DIPY.

Throughout the years, many tools were developed in scilpy before being mature enough to be moved into DIPY (e.g., the full spherical harmonics [SH] basis, the asymmetric peak direction, the Bingham model, the Stateful Tractogram, the option to save seeds in .trk files, and so on). To compare the current state of both libraries, we can discuss scripts and Python functions. Regarding scripts, DIPY offers workflows accessible in command-line interface (CLI), similar to our scripts. In particular, they offer options for local reconstruction methods and the tractography algorithm. In those cases, our scripts allow access to the same core functions, sometimes with added detailed documentation. But we also have other scripts that we created to answer needs in the community (see section 5), and we offer many post-tractography analysis scripts. Some scripts are 100% original, others rely more heavily on DIPY, but even in those cases, we complement DIPY by offering management of default values, management of data loading and saving, verifications of input requirements (e.g., header compatibility across files), and often management of computer resources to allow multi-processing. In Supplementary Table A2, we list the DIPY functions accessed in our scripts. Regarding functions, both scilpy and DIPY are mainly Python libraries, and we ensure that our libraries are compatible. For instance, scilpy uses DIPY's data object structures (e.g., Stateful Tractograms).

The second important comparison is between scilpy and other libraries. In Table 1, we describe the strengths and limitations of scilpy for each category of scripts. In general, scilpy particularly excels in tractogram and bundle operations. We only offer a few options for dMRI volume denoising, segmentation and registration, but other libraries already offer many strong tools (e.g., FSL,[14] ANTs[15] and others). Similarly, we offer a few visualization scripts but no real complex interactive visualization tools. Other libraries already offer good options (e.g., MI-Brain, MRtrix's mrview[8] or FSL's fslview[14]). MRtrix[8] should be mentioned in particular as a similar library to scilpy. It also offers local reconstruction, tractography, and connectomics analysis, but it currently does not support tractometry or profilometry analyses. It does, however, provide options for fixel analysis. We cannot provide a thorough efficiency comparison (computer resource requirements, processing time, quality of the results) because this would require testing

Table 1. Description of the wide range of scripts available in the scilpy library. Scripts in scilpy are separated into categories, showing the possibilities offered by the library. Note that all scilpy's scripts follow the scil_{category}_{description} pattern (for instance, scil_tracking_local), for a quick understanding of the goal of the script. This also allows a smooth browsing experience of scripts: in a terminal, entering scil_{category}_ (for instance, scil_tracking_) and hitting tab twice lists all scripts available in that category. N: Number of scripts in version 2.2.1.

Category	Category prefix	N	Explanation	Comparison to other libraries and tools[a]
Volume utilities (3D or 4D)	volume labels	14 6	Simple operations on volumes (resampling, flipping axes, cropping, mathematical operations, etc.) and operations on labels (dilation, erosion, label selection, etc).	Basic operations that can also be done by other libraries such as AFNI, (e.g., 3dcalc), MRtrix3 (e.g., mrcalc), FSL (e.g., fslmath), including registration or segmentation offered by ANTs, FreeSurfer, AFNI, MRtrix3, DIPY, etc.
dMRI utilities	dwi gradients	13 9	Simple operations on diffusion-weighted imaging (DWI) data accompanied by its b-values and b-vectors files, such as shell extraction, outlier detection, etc. Management of b-values and b-vectors.	Some of these operations can also be done by dMRI libraries, particularly MRtrix3, and by other libraries to a lesser extent (AFNI, FSL, Tortoise, DIPY, or Freesurfer).
denoising	denoising	1	Access to DIPY's NL-means denoising.	Other libraries already offer complete options.
dMRI local reconstruction methods	dti fodf frf and more	32	Local reconstruction methods for microstructural properties and orientational information, using tensors, fODF and asymmetric fODF, b-tensors, freewater, NODDI, qball, and more.	For single-shell or multi-shell CSD, MRtrix3 and scilpy offer similar options. MRtrix3 provides more flexibility for fiber response function estimation. Scilpy stands out for its wider array of local diffusion methods like DKI, MEMS-MT (for multi-encoding, multi-shell, and multi-tissue), freewater, and NODDI, thanks to its interface with DIPY.
Tractography	tracking	5	Local or PFT tracking (deterministic or probabilistic), based on tensor, fODF, or peak inputs. Scilpy's official tracking script matches DIPY's code's speed (based on Cython). * Scilpy also offers PFT maps creation and manipulation scripts. * We also offer an all-Python option, slower but easier to adapt to any user's needs when testing new features. Better adapted for educational purposes and internship students.	Scilpy, MRtrix3, and DSI Studio have comparable basic tractography options. However, scilpy includes various algorithms (EuDx, PTT, PFT). Scilpy also offers a GPU-accelerated version of local (probabilistic or deterministic) tractography for faster processing.
Tractogram operations	tractogram	41	Operations on the tractogram, such as streamline filtering, resampling of the number of points per line, resampling of the number of streamlines in the tractogram, segmentation into bundles (from ROI inclusion/exclusion, or with Recobundles / Bundleseg), segmentation into connections for connectomics projects, tractogram registration, and tractogram operations such as concatenation, intersection, difference, etc.	MRtrix3 offers some options (e.g., tckresample, tcktransform), but, to our knowledge, no other library offers such a wide range of possibilities as scilpy to process streamlines and tractograms. pyAFQ[18] and phybers[19] offer python functions for bundle recognition, but no script.
Bundle operations	bundle json	24	Operations on tractograms that are known to represent a single pathway, or a single bundle, such as filtering, comparison to a reference, statistics (e.g., tractometry) or fixel analysis.	Such operations and statistics are rarely offered in other libraries. pyAFQ[18] offers python functions for tract profiling, but no script.
Connectomics	connectivity	12	Computation of connectivity matrices, based on various weights. Operations and statistics on matrices such as row/columns reordering, matrix normalization, population comparison.	MRtrix3 offers connectivity matrix computation (tck2connectome). Graph theory measures are available in MATLAB using the Brain Connectivity Toolbox.[20]
Visualization	viz	16	Creation of plots for some statistics based on dMRI data. Creation of mosaics of images to visualize bundles. Visualization of fODFs, streamlines, streamtubes, connectivity matrices, circle graphs (also called chord charts), and more.	Other libraries offer more complete 3D visualization tools, such as AFNI (afni), FreeSurfer (Freeview), MRtrix3 (mrview), FSL (fsleyes), etc. However, few of them offer streamline visualization. MI-Brain and TrackVis[21] can be named as particularly useful for this task, but are no longer maintained.
Other	Surface and more	2 10	Scilpy offers utility scripts, for instance to search information throughout scripts, to read and validate headers (from NIfTI or TRK files) or BIDS structures, or to convert between formats. Generally, our scripts accept NIfTI formats for volume images, and TRK and TCK formats for tractograms. Scilpy scripts support surface operations. An option to segment bundle maps directly from the fODF, without requiring tractography data, has been recently added. A few scripts allowing to analyse non-healthy subjects' data have been created upon requests from collaborators.	

[a] Amongst the most established libraries, MRtrix3 is a good example of a command-line interface-oriented (CLI) software; DSI-Studio offers CLI options but is mainly known for its graphical user interface (GUI); and DIPY's Python API is extensively used, but has fewer workflows available through CLI. Other libraries, such as FreeSurfer or ANTs, are named here as they have become essential for more specialised tasks, including segmentation and registration.

all options of all scripts, and comparing results against a ground truth, which is a known limitation in our field.[22] But it is worth mentioning that their tractography script runs faster. This is a consequence of the choice of language (MRtrix uses C++), but we believe that using Python helps in making dMRI accessible to newcomers for educational purposes (see section 6).

We note that scilpy supports many data formats to allow compatibility with other libraries and offers scripts to convert volumes or tractogram files.

Scilpy fills a need in the community. Examples of the usefulness of scilpy already appear in the literature. Bundle filtering tools have been used to segment and score bundles in comparison to a reference[23] and to analyse WM in aging[24]; tractometry (and profilometry) tools have allowed investigations of tau pathology[25]; and connectomics tools have been used to analyse connectivity alterations in epilepsy.[26]

SECTION 3: INCLUDING SCILPY INTO SCIENTIFIC PROJECTS

An important aspect of scilpy is its emphasis on ensuring that each script performs a single, well-defined task. This granularity leads to a very high number of scripts but allows users to manipulate data exactly as they want. To organize processing steps in their favorite order or to automate processes, for a single subject or for larger databases, users may use their favorite technology. Options range from simply aligning command lines sequentially in a bash script to using more complex workflow management systems such as Nextflow,[27] Nipype,[28] snakemake,[29] and more, which allow parallel processing over various subjects for accelerated workflows. nf-neuro (https://github.com/nf-neuro) is our team's take on a Nextflow-based pipeline and will be the subject of an upcoming publication. Users interested in processing large cohorts may also discover Tractoflow,[30] which includes 26 scilpy scripts and has already led to the creation of multiple-subject databases of processed data.[31, 32]

We note that the efficiency of pipelines often depends on a clear database organization. Again, this is out of the scope of scilpy, and interested users may find instructions for best practices in other references, such as BIDS.[33]

In sections below, we showcase a few examples of analysis objectives and the sequence of scripts required to achieve them. The bash scripts associated with each task are available in the online tutorials.

OBJECTIVE 1: FROM RAW DATA TO DTI, FODF, AND FULL TRACTOGRAM

Scilpy scripts allow the complete processing of raw dMRI data in NIfTI format to obtain a full tractogram, such as shown on Figure 1. A complete tractography process can be separated into subsections including 1) a general organization of data, 2) preprocessing and denoising of the DWI data, 3) computation of the local representation, 4) tractography, and 5) filtering out improbable streamlines.

1) During data organization, the user prepares the b-values and b-tensors text files, extracts the b0 and/or the reversed-b0 from the DWI using scil_dwi_extract_b0, and may choose to modify the DWI file in order to improve and facilitate the next processing steps with scripts such as scil_volume_crop, scil_volume_resample, or scil_dwi_concatenate.

2) Preprocessing of the DWI can include rejecting problematic volumes with scil_dwi_detect_volume_outliers or denoising with scil_denoising_nlmeans. The user may include other preprocessing steps of their choice, such as registration to the MNI template with ANTS or skull-stripping with FSL. The computation of the local representation requires the user to first convert the raw signal to SH using scil_dwi_to_sh.

3) Scilpy has scripts to compute various local reconstruction methods (scil_dti_metrics, scil_dki_metrics, scil_qball_metrics, scil_freewater_maps, scil_NODDI_maps),[1,34-37] particularly many variations of fODFs (scil_fodf_metrics, scil_fodf_msmt, scil_fodf_memsmt, scil_fodf_ssst).[38] Figure 1 shows an example using fODFs as the local representation, while also computing DTI to obtain crucial information such as the FA map. At this stage, the user may use these results for further analysis such as fixel analysis (scil_bingham_metrics).[39]

4) The next step is the tractography. Our main tractography script, scil_tracking_local, gives access to all of DIPY's efficient Cython-based tracking options, and more, such as particle filtering (PFT), parallel transport (PTT) and Euler deterministic (EuDx). Alternatively, scil_tracking_local_dev uses a slower Python loop, but is closer to how one would teach tractography to a newcomer, and allows easy testing of improvement ideas for new developers. Furthermore, this script offers an accelerated GPU-based option or multiprocessing on CPU. Finally, the user could choose to compare results with any other library's tractography. In particular, scilpy allows saving the fODFs in a SH basis, compatible with MRtrix3 to use their tractography, and conversely, it can perform tractography on fODFs from MRtrix3.

5) The resulting tractogram usually requires postprocessing steps, such as the manipulations described in Objectives #2 - #4. The user could also choose to continue their pipeline with other libraries: our tractography scripts enable saving tractograms as either TRK or TCK to facilitate interoperability with any software accepting these standard formats.

OBJECTIVE #2: TRACTOGRAM MANIPULATIONS

Scilpy scripts allow users to modify tractograms in various ways based on their needs, such as shown on Figure 2.

Some scripts allow operations on the tractogram as a whole object in the brain, such as flipping it on a chosen axis (scil_tractogram_flip) or creating a map of all voxels touched by a streamline (scil_tractogram_compute_density_map). Mathematical operations on two tractograms such as union, intersection, and difference can be performed through the scil_tractogram_math script.

Other scripts allow operations on the tractogram as a set of streamlines, such as resampling the number of stream-

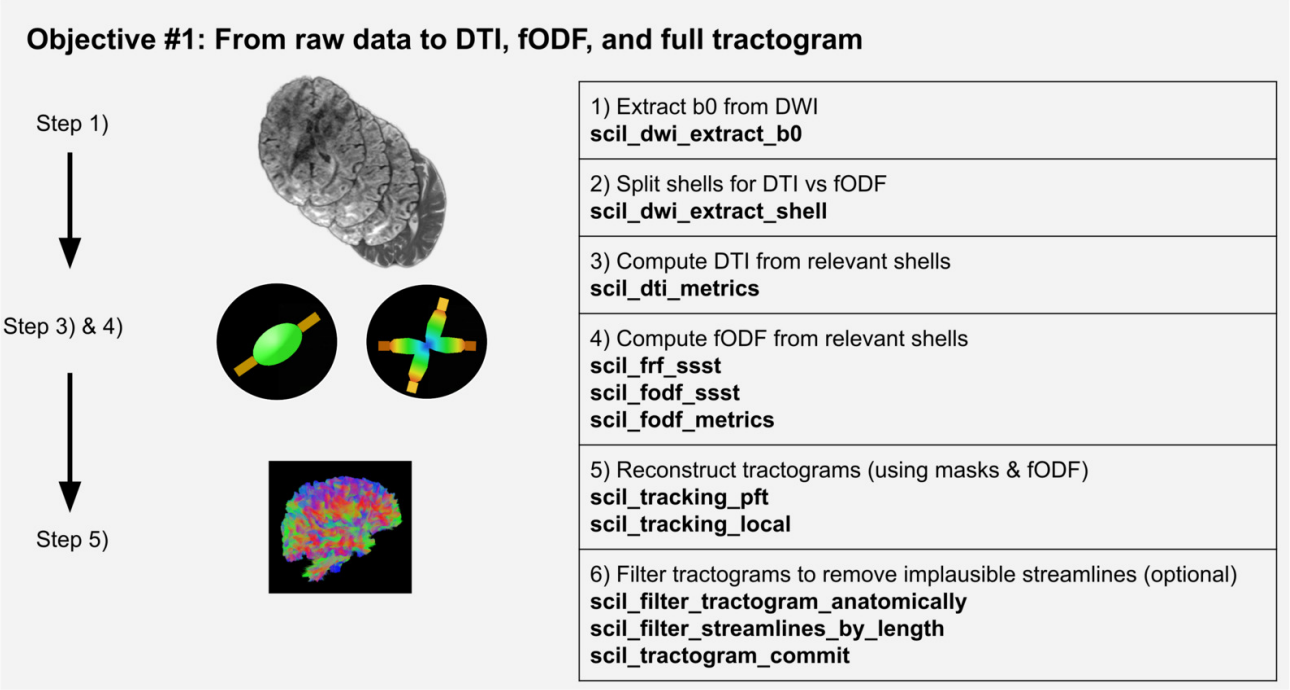

Figure 1. Sequence of scripts to generate a full tractogram from raw data (Objective #1).

Simplified example of a pipeline used to generate a full tractogram from raw DWI data in NIfTI format. This pipeline can be expanded with more advanced preprocessing and reconstruction options. For instance, volumes can be cropped or resampled, and DWI data can be concatenated or checked for outliers before reconstruction. Beyond the DTI and single-shell fODF models shown, scilpy supports various local reconstruction methods, including diffusion kurtosis imaging (DKI), multi-shell and multi-tissue fODF models, NODDI, and q-ball ODFs, allowing for a more detailed characterization of the tissue microstructure.

lines (scil_tractogram_resample, scil_tractogram_split), separating streamlines based on various criteria (scil_tractogram_filter_by_roi, scil_tractogram_filter_by_anatomy, scil_tractogram_filter_by_length, scil_tractogram_filter_by_orientation) or segmenting a tractogram into bundles (see Objective #3).

Finally, other scripts allow modifying the streamlines themselves, for instance by resampling the number of points on each streamline (scil_tractogram_resample_nb_points, scil_tractogram_compress), or smoothing the streamlines' trajectories (scil_tractogram_smooth).

Furthermore, when using the .trk format, additional information may be stored in memory for each streamline (data_per_streamline [dps]) or for each individual point (data_per_point [dpp]). It is possible, for instance, to store the seeding point of each streamline as dps during the tractography and use it later to create a map of all seeding points leading successfully to a streamline (scil_tractogram_seed_density_map). It is also possible to store a RGB color to each point as dpp, associated to the keyword 'color', which allows to visualise the tractogram as desired on a visualization software (e.g., Mi-Brain[40] supports the 'color' keyword). Finally, is it possible to perform more complex actions when analysing some metrics of interest along the streamlines, as described in Objective #4.

OBJECTIVE #3: CREATING A WM BUNDLE POPULATION TEMPLATE AND VISUALIZING IT

Scilpy scripts enable users to create a WM bundle population template (see Figure 3). Such a pipeline includes 1) segmenting the bundle of interest in each subject's tractogram, 2) registering the bundles to a reference space (e.g., MNI space) and analysing the inter-subject variability, 3) combining them into a reference bundle template, similar to how one would average many structural MRI images to create a brain template, and 4) analysing the results.

1) The segmentation of bundles can be based on ROIs of inclusion or exclusion (scil_tractogram_segment_with_ROI_and_score or scil_tractogram_filter_by_roi) or based on the general shape of the streamlines (scil_tractogram_segment_with_recobundles, scil_tractogram_segment_with_bundleseg).[41,42] The resulting bundles can be cleaned more thoroughly than whole-brain tractograms, considering that their shapes should be quite uniform, and spurious streamlines can be discarded automatically with scil_bundle_reject_outliers or visually with scil_bundle_clean_qbx_clusters. The bundle could also be cut or trimmed using a binary mask, allowing it to focus on a specific region, using scil_tractogram_cut_streamlines. Optionally, metrics and statistics can be measured for each individual subject using our various scripts for bundle analysis, such as presented in Objective #4.

2) Then, registration can be performed by computing the registration matrix between the subjects' space and the reference space (e.g., using ANTS) and applying the same registration matrix to the tractogram's coordinates with scil_tractogram_apply_transform. Inter-subject variability can be measured with scil_tractogram_pairwise_comparison.

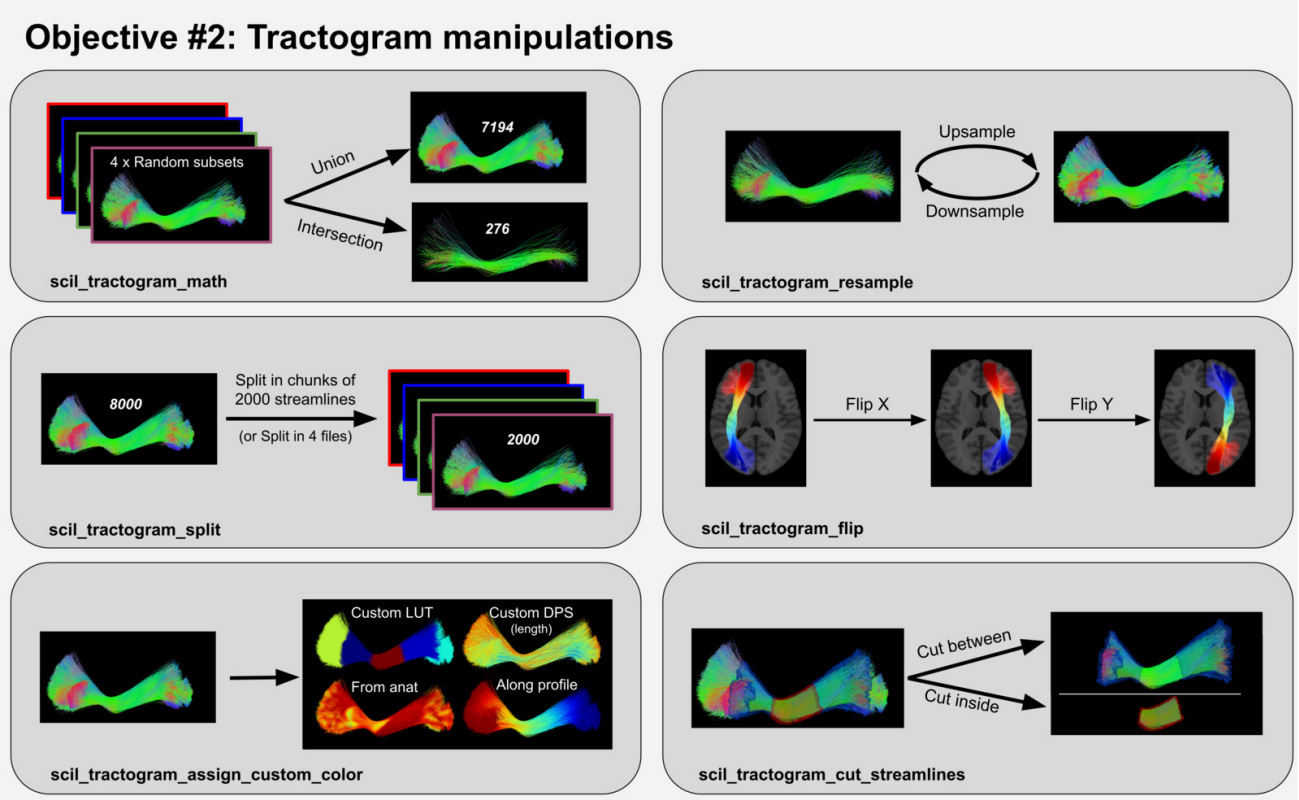

Figure 2. Examples of tractogram manipulations (Objective #2).

Scilpy scripts allow modifying tractograms (bundles or whole brain) in various ways. Since our scripts do simple operations, they can easily be chained together to achieve complex tasks.

3) The tractograms can be downsampled and concatenated, as described in Objective #2, and even concatenated to their flipped version to obtain a symmetrical template.

OBJECTIVE #4: SEGMENTING AND CLEANING A BUNDLE, AND PERFORMING PROFILOMETRY

Using a clean bundle from a single subject, scilpy allows performing a profilometry[5] analysis, which is the analysis of the evolution of any dMRI metric along its subsections. A simple example is shown in Figure 4, and for further examples of such processes, users can refer to Tractometry-flow (https://github.com/scilus/tractometry_flow), a Nextflow process using many scilpy scripts, or to the study by Cousineau et. al.[43] Such a pipeline includes the same subsections as in Objective #3 (segmentation and cleaning of the bundles of interest in each subject), and analysis of the resulting bundles.

The profilometry analysis requires segmenting the bundles into as many subsections as desired. This is not straightforward, as the division of sections can be performed arbitrarily. We generally cut the section perpendicularly to the direction of the bundle, measured from a centroid, a single streamline-like shape representing the average of all streamlines in the bundle (scil_bundle_compute_centroid, scil_bundle_label_map). Then, any metric can be associated with the subsections. Figure 4 shows the example of the sections' volume and diameter, or mean underlying value of any given map, such as the FA map. The resulting .json file's values can be plotted with scil_plot_stats_per_point.

Beyond profilometry, other analysis steps could also be performed on the whole bundle. For instance, it is possible to compute the map of the head and tail of a bundle (the starting and ending regions of the bundle, assuming all streamlines are aligned in the same direction) with scil_bundle_uniformize_endpoints and scil_bundle_compute_endpoints_map. It is also possible to extract various information with scil_bundle_shape_measures.

OBJECTIVE #5: ANALYSING CONNECTIVITY

Connectivity analyses use whole-brain tractograms segmented based on a gray matter (GM) parcellation to create various connectivity matrices (see Figure 5).

For a very simple computation, the GM label associated to each endpoint of the streamlines can be used to create a connectivity matrix based on streamline count with scil_connectivity_compute_simple_matrix. However, scilpy also offers a more thorough analysis, with a more complex analysis of endpoints and more options for the weight of the connectivity matrix. It comprises the steps described below.

First, the script scil_tractogram_segment_connections_from_labels is used and results in many sub-bundles, one for each pair of GM regions (each pair of labels). This is similar to ROI-based segmentation, but its computation makes a more complex analysis in cases where streamlines

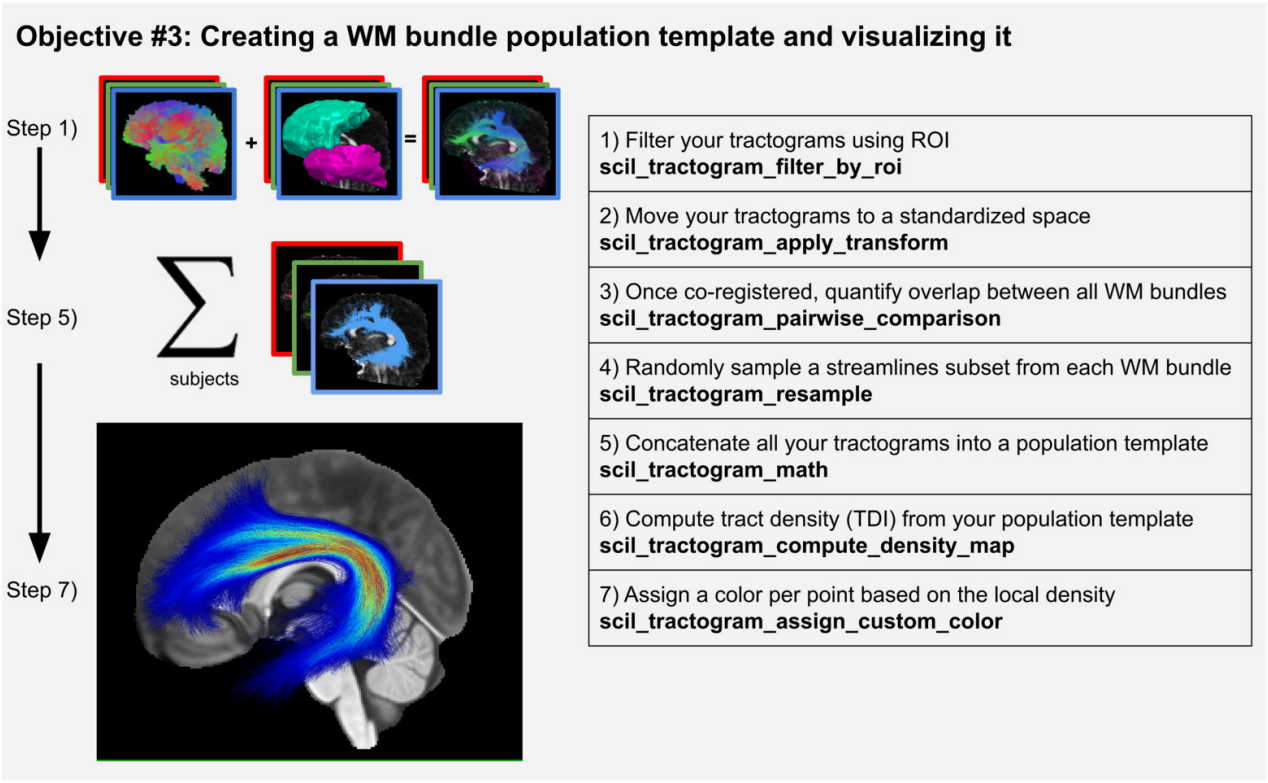

Figure 3. Sequence of scripts for the creation and visualization of a template bundle (Objective #3).

Scilpy scripts allow creating a reference bundle template for a given population. The example above uses a ROI-based segmentation, but other options are available. Once the bundles are clean and registered to a common space, they are downsampled to save space before concatenating all subjects together and performing statistical analyses. It is possible to visualise the inter-subject variability by loading individual tractograms in a visualization software, or to view the spatial range of the population template and its density in each voxel.

cross many GM regions. The output is saved in the HDF5 format for the ease of use in subsequent scripts but can be converted back to a list of .trk files with scil_tractogram_convert_hdf5_to_trk. Any manipulation can be performed on the bundles and then they could be merged back (scil_tractogram_convert_trk_to_hdf5). Note that bundle registration and analysis presented before could also be computed directly from the resulting HDF5 format, such as warping (scil_tractogram_apply_transform_to_hdf5) or statistics computation (scil_bundle_mean_fixel_afd_from_hdf5).

Then, it is possible to compute the connectivity matrix, using scil_connectivity_compute_matrices, using many weights such as the streamline count, the lengths of streamlines, the volume of bundles, or the average of any underlying map or any dps/dpp stored in the data.

It is possible to modify the matrices afterward with scripts such as scil_connectivity_math (operations such as threshold, addition, multiplication, interchanging rows, etc.), scil_connectivity_filter (to binarize a list of matrices based on conditions) or scil_connectivity_normalize (to modify the minimum and maximum values).

SECTION 4: ORGANIZATION OF FUNCTIONS, TESTING, AND MAINTENANCE

By using Python as the coding language, scilpy benefits from integration with widely used and well-tested open-source packages such as NumPy[46] (array operations), SciPy[47] (scientific computing), NiBabel[48] (neuroimaging file management), and DIPY[7] (denoising, white matter local reconstructions, tractography algorithms, bundle analysis and spatial transform management of tractograms).

In scilpy, the functions are organized into clearly defined modules (Python files), making it easy to understand the library's structure. Separation of functions into modules generally follows the separation of scripts into categories (see Table 1). Scilpy's modular structure allows users to reuse and integrate the functions in scilpy in their projects effortlessly.

Scilpy has rigorous testing code through unit tests and smoke tests in order to ensure its robustness and reproducibility. Smoke tests are minimal tests ensuring that all options in scripts are properly handled and that all scripts can run from start to finish without error. As of October 2025 (version 2.2.1), code coverage covered by smoke tests is 70%. Our goal is to reach approximately 75% coverage for smoke tests. Unit tests ensure that the results of individual functions are as expected, based on numerical results. If analytical testing is not possible, scilpy's unit tests confirm that the resulting output values remain consistent across

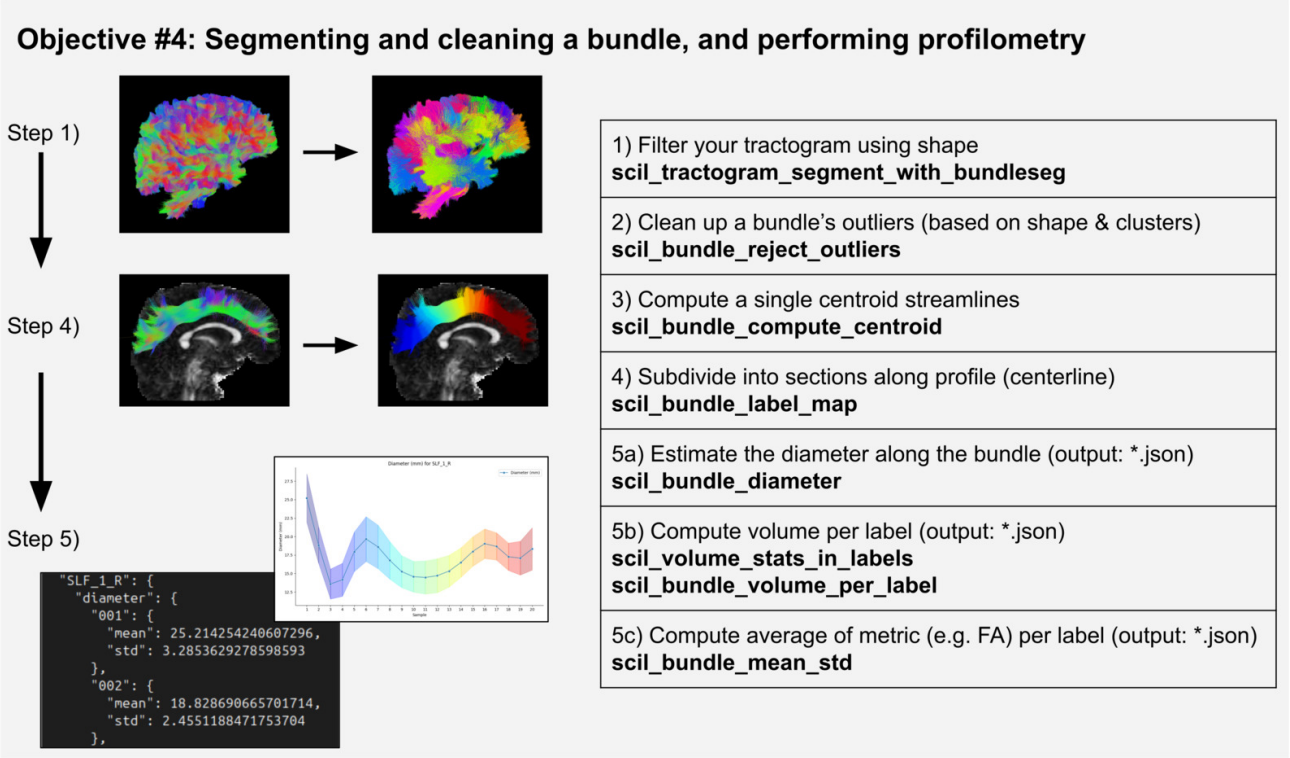

Figure 4. Sequence of scripts allowing to perform profilometry (Objective #4).

This sequence of scripts allows performing profilometry for a bundle extracted with BundleSeg.[44] Here, a single cleaning step is performed (step 2), using a hierarchical clustering method.[45] In steps 3 and 4, the bundle is subdivided into sections (using a centroid to determine a Euclidean association). Finally, step 5 allows quantifying each section of the bundle (shape, volume, metrics, etc.). The JSON output allows easy plotting using any library or conversion to Excel spreadsheets.

versions. Current coverage by unit tests is at 29% for the library codebase. These tests were added recently to the project, in response to scilpy's increasing recognition and usage, and unit test development is ongoing. Our goal is to reach 100% of our main functions, meaning functions that use DIPY sparsely (DIPY already tests its functions thoroughly) and functions that are not used for visualization, which is difficult to test actively. Modules lacking smoke tests are visualization (cannot be tested automatically) and machine learning (recently added to scilpy). Modules lacking unit tests are post-tractography modules, but, again, development is ongoing.

SECTION 5: EASE OF COLLABORATION USING SCILPY

Scilpy's development team has a vision that supports the rapid implementation of new ideas. Scripts are primarily designed to meet user needs.

For example, a common development scenario starts with a collaborator with limited or no computer programming background requiring a specific feature. For instance, they might request a coloring tool where each streamline is colored according to its minimum distance to any streamline within a template, using a designated colormap. This request prompts a discussion regarding input and output data, the required reference frame for the operation, and performance considerations (speed, robustness, and ease of use). The design principles include having the minimal set of arguments necessary towards both collaborator satisfaction and script reusability. The development process begins with a proof-of-concept implementation, which is then shared with the collaborator for iterative refinement, bug reporting, and improvements to the documentation. A context-agnostic development team member reviews the new script to simulate the experience of a new user, emphasizing usability and operational behaviour. Subsequent reviews address code optimization, modularity, input/output handling, general code quality, and implementing unit tests and smoke tests. This entire process typically takes a few weeks. Pull requests are generally merged in a timely fashion and usability can be further enhanced over time as the script is reused and refined based on feedback from subsequent collaborator requests.

The objective of scilpy is to have regular releases. This does require a soft reviewing style but ensures the quick inclusion of collaborative work to our system. On the contrary, DIPY works with a stronger reviewing process and more rigorous code style, resulting in a slower release frequency. Such stricter requirements are necessary due to DIPY's more widespread use and its functions being used as a foundation for hundreds of projects in multiple labs across the world, and the need to ensure stability for the users. In contrast, the faster release frequency in scilpy helps developers adapt better to rapidly evolving scientific and clinical research projects. Scripts that are not yet fully tested have disclaimers in their description.

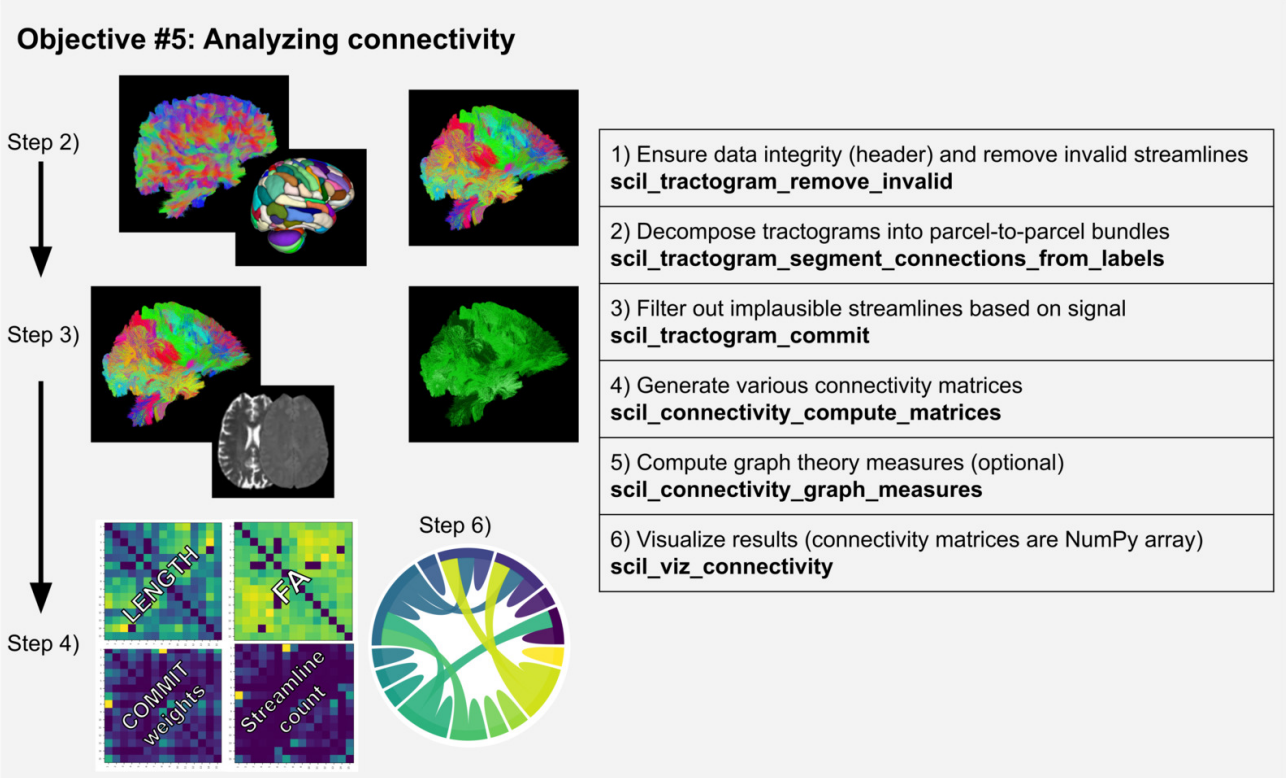

Figure 5. Connectivity analysis (Objective #5).

Connectivity analysis uses parcel-to-parcel bundles, computed at step 2. After step 4, various connectivity matrices are computed. Steps 5 and 6 are useful to obtain statistics from the matrix.

SECTION 6: SCILPY FOR EDUCATIONAL PURPOSES

Scilpy has been designed with a strong educational component in mind. Firstly, it allows research students and collaborators to quickly become familiar with the implementation of complex theoretical aspects in dMRI and MRI data management concepts (3D/4D volumes, tractograms, etc.) by providing a rich set of code examples with integrated input/output from common toolboxes. The use of Python as a programming language also facilitates contributions due to its ease-of-use. Cutting-edge features can mature and survive their contributors by being integrated early into scilpy. Secondly, scilpy's audience (graduate research students, scientific and clinical researchers, computational scientists, and self-paced curious individuals) can leverage years of expertise in dMRI, embedded into the default parameters, to quickly develop a prototype of their ideal/needed processing pipeline, regardless of their proficiency level. Finally, scilpy is a core component of the SCIL, acting as both a practical tool and a reference implementation for members and collaborators. It embodies and promotes good coding practices and open-source principles, helping to advance the field of dMRI and tractography while raising the standards of the scientific contributions it supports in terms of robustness, reproducibility, replicability, and generalizability. These three educational components help increase scilpy's educational, scientific, and academic impact.

SECTION 7: LIMITATIONS AND FUTURE PERSPECTIVES

We expect scilpy to continue evolving based on collaborations and research projects at the SCIL. Having high-quality documentation and examples are key to its adoption and success, and we intend to build a set of tutorials in the near future. Training the next generation of researchers is important, and we intend to offer workshops on dMRI data processing using scilpy tools and others. We aim to attract new developers to contribute to scilpy through its public GitHub repository.

Although many scripts already offer multiprocessing options, current limits in scilpy's scripts include RAM management and optimization. We continue actively improving our scripts with every new collaboration. Another limitation from our choice of organization, is that having single-step scripts inevitably creates many intermediate files that would not necessarily be stored on disk with more pipeline-like scripts doing multiple steps at once. This effect can be mitigated with good processing practices.

CONCLUSION

Scilpy offers unique scientific contributions, addresses critical needs in the computational neuroimaging field, and exemplifies the diverse impacts that open science can have on academic research. Scilpy will help the dMRI scientific com-

munity achieve its goals by accelerating the data processing, with a wide range of well documented scripts that cover many scientific and clinical researchers' needs, and with an easy development process through open-source Python programming.

DATA AND CODE AVAILABILITY

All code is available publicly on GitHub or PyPI.

ACKNOWLEDGEMENTS

Thanks to Yuzhe Wang, Hermela Gebremariam, Élise Cosenza, Florence Gagnon and Alexandre Gauvin for their contribution in the Github repository. We would like to thank Kurt Schilling for his useful comments on the text.

FUNDING SOURCES

Authors MD and FR thank their respective NSERC Discovery grants (RGPIN-202004818) and the Université de Sherbrooke institutional research chair in neuroinformatics.

CONFLICTS OF INTEREST

MD is co-founder and shareholder at Imeka Solutions. He has no conflict of interest with respect to the content of this manuscript. All other authors have no conflicts of interests to declare.

Submitted: July 21, 2025 CST. Accepted: November 20, 2025 CST. Published: January 08, 2026 CST.

REFERENCES

1. Pierpaoli C, Jezzard P, Basser PJ, Barnett A, Di Chiro G. Diffusion tensor MR imaging of the human brain. *Radiology*. 1996;201(3):637-648. doi:10.1148/radiology.201.3.8939209

2. Dell'Acqua F, Tournier JD. Modelling white matter with spherical deconvolution: How and why? *NMR Biomed*. 2019;32(4):e3945. doi:10.1002/nbm.3945

3. Jeurissen B, Descoteaux M, Mori S, Leemans A. Diffusion MRI fiber tractography of the brain. *NMR Biomed*. 2019;32(4):e3785. doi:10.1002/nbm.3785

4. Catani M, Howard RJ, Pajevic S, Jones DK. Virtual in Vivo Interactive Dissection of White Matter Fasciculi in the Human Brain. *NeuroImage*. 2002;17(1):77-94. doi:10.1006/nimg.2002.1136

5. Yeatman JD, Dougherty RF, Myall NJ, Wandell BA, Feldman HM. Tract Profiles of White Matter Properties: Automating Fiber-Tract Quantification. *PLOS ONE*. 2012;7(11):e49790. doi:10.1371/journal.pone.0049790

6. Sotiropoulos SN, Zalesky A. Building connectomes using diffusion MRI: why, how and but. *NMR Biomed*. 2019;32(4):1-23. doi:10.1002/nbm.3752

7. Garyfallidis E, Brett M. Dipy, a library for the analysis of diffusion MRI data. *Front Neuroinformatics*. 2014;8:1-14. doi:10.3389/fninf.2014.00008. PMID:24600385

8. Tournier JD, Smith R, Raffelt D, et al. MRtrix3: A fast, flexible and open software framework for medical image processing and visualisation. *NeuroImage*. 2019;202:116137. doi:10.1016/j.neuroimage.2019.116137

9. Irfanoglu MO, Nayak A, Jenkins J, Pierpaoli C. TORTOISE v3: Improvements and New Features of the NIH Diffusion MRI Processing Pipeline. *Program Proc ISMRM 25th Annu Meet Exhib Honol HI USA*. Published online April 2017.

10. Yeh FC. Population-based tract-to-region connectome of the human brain and its hierarchical topology. *Nat Commun*. 2022;13(1):4933. doi:10.1038/s41467-022-32595-4

11. González Rodríguez LL, Osorio I, Cofre GA, et al. Phybers: a package for brain tractography analysis. *Front Neurosci*. 2024;18. doi:10.3389/fnins.2024.1333243

12. Dale AM, Fischl B, Sereno MI. Cortical surface-based Analysis: I. segmentation and surface reconstruction. *NeuroImage*. 1999;9(2):179-194. doi:10.1006/nimg.1998.0395

13. Cox RW. AFNI: software for analysis and visualization of functional magnetic resonance neuroimages. *Comput Biomed Res Int J*. 1996;29:162-173. doi:10.1006/cbmr.1996.0014

14. Jenkinson M, Beckmann CF, Behrens TEJ, Woolrich MW, Smith SM. FSL. *NeuroImage*. 2012;62:782-790. doi:10.1016/j.neuroimage.2011.09.015

15. Avants B, Tustison N, Song G. Advanced Normalization Tools (ANTS). *Insight J*. Published online 2009:1-35. doi:10.54294/uvnhin

16. Gagnon A, Grenier G, Bocti C, et al. White matter microstructural variability linked to differential attentional skills and impulsive behavior in a pediatric population. *Cereb Cortex*. 2023;33(5):1895-1912. doi:10.1093/cercor/bhac180

17. Roy M, Rheault F, Croteau E, et al. Fascicle- and Glucose-Specific Deterioration in White Matter Energy Supply in Alzheimer's Disease. *J Alzheimer's Dis*. 2020;76(3):863-881. doi:10.3233/JAD-200213

18. Kruper J, Yeatman JD, Richie-Halford A, et al. Evaluating the reliability of human brain white matter tractometry. *Apert Neuro*. 2021;1(1). doi:10.52294/e6198273-b8e3-4b63-babb-6e6b0da10669. PMID:35079748

19. Guevara P, Duclap D, Poupon C, et al. Automatic fiber bundle segmentation in massive tractography datasets using a multi-subject bundle atlas. *NeuroImage*. 2012;61(4):10831099. doi:10.1016/j.neuroimage.2012.02.071

20. Rubinov M, Sporns O. Complex network measures of brain connectivity: Uses and interpretations. *NeuroImage*. 2010;52(3):1059-1069. doi:10.1016/j.neuroimage.2009.10.003

21. Wang R, Wedeen VJ. TrackVis.org. Martinos Center for Biomedical Imaging, Massachusetts General Hospital.

22. Drobnjak I, Neher P, Poupon C, Sarwar T. Physical and digital phantoms for validating tractography and assessing artifacts. *NeuroImage*. 2021;245. doi:10.1016/j.neuroimage.2021.118704

23. Renauld E, Théberge A, Petit L, Houde JC, Descoteaux M. Validate your white matter tractography algorithms with a reappraised ISMRM 2015 Tractography Challenge scoring system. *Sci Rep*. 2023;13(1):2347. doi:10.1038/s41598-023-28560-w

24. Schilling KG, Archer D, Yeh FC, et al. Short superficial white matter and aging: A longitudinal multi-site study of 1293 subjects and 2711 sessions. *Aging Brain*. 2023;3:100067. doi:10.1016/j.nbas.2023.100067

25. Qiu T, Liu ZQ, Rheault F, et al. Structural white matter properties and cognitive resilience to tau pathology. *Alzheimers Dement*. doi:10.1002/alz.13776

26. Obaid S, Rheault F, Edde M, et al. Structural Connectivity Alterations in Operculo-Insular Epilepsy. *Brain Sci*. 2021;11(8):1041. doi:10.3390/brainsci11081041

27. Di Tommaso P, Chatzou M, Floden EW, Barja PP, Palumbo E, Notredame C. Nextflow enables reproducible computational workflows. *Nat Biotechnol*. 2017;35(4):316-319. doi:10.1038/nbt.3820

28. Gorgolewski K, Burns CD, Madison C, et al. Nipype: A Flexible, Lightweight and Extensible Neuroimaging Data Processing Framework in Python. *Front Neuroinformatics*. 2011;5. doi:10.3389/fninf.2011.00013

29. Mölder F, Jablonski KP, Letcher B, et al. Sustainable data analysis with Snakemake. *F1000Research*. 2025;10:33. doi:10.12688/f1000research.29032.3

30. Theaud G, Houde J christophe, Bor A, Morency F, Descoteaux M. TractoFlow : A robust, efficient and reproducible diffusion MRI pipeline leveraging Nextflow & Singularity. *NeuroImage*. 2020;218(March). doi:10.1016/j.neuroimage.2020.116889

31. Poulin P, Theaud G, Rheault F, et al. TractoInferno - A large-scale, open-source, multisite database for machine learning dMRI tractography. *Sci Data*. 2022;9(1):725. doi:10.1038/s41597-022-01833-1

32. Edde M, Theaud G, Dumont M, et al. High-frequency longitudinal white matter diffusion- and myelin-based MRI database: Reliability and variability. *Hum Brain Mapp*. 2023;44(9):3758-3780. doi:10.1002/hbm.26310

33. Gorgolewski KJ, Alfaro-Almagro F, Auer T, et al. BIDS apps: Improving ease of use, accessibility, and reproducibility of neuroimaging data analysis methods. *PLOS Comput Biol*. 2017;13(3):e1005209. doi:10.1371/journal.pcbi.1005209

34. Jensen JH, Helpern JA, Ramani A, Lu H, Kaczynski K. Diffusional kurtosis imaging: The quantification of non-gaussian water diffusion by means of magnetic resonance imaging. *Magn Reson Med*. 2005;53(6):1432-1440. doi:10.1002/mrm.20508

35. Descoteaux M, Angelino E, Fitzgibbons S, Deriche R. Regularized, fast, and robust analytical Q-ball imaging. *Magn Reson Med*. 2007;58(3):497-510. doi:10.1002/mrm.21277

36. Pasternak O, Sochen N, Gur Y, Intrator N, Assaf Y. Free water elimination and mapping from diffusion MRI. *Magnetic Resonance in Medicine: An Official Journal of the International Society for Magnetic Resonance in Medicine*. Published online 2009:717-730. doi:10.1002/mrm.22055

37. Zhang H, Schneider T, Wheeler-Kingshott CA, Alexander DC. NODDI: Practical in vivo neurite orientation dispersion and density imaging of the human brain. *NeuroImage*. 2012;61(4):1000-1016. doi:10.1016/j.neuroimage.2012.03.072

38. Raffelt D, Tournier JD, Rose S, et al. Apparent Fibre Density: A novel measure for the analysis of diffusion-weighted magnetic resonance images. *NeuroImage*. 2012;59(4):39763994. doi:10.1016/j.neuroimage.2011.10.045

39. Riffert TW, Schreiber J, Anwander A, Knösche TR. Beyond fractional anisotropy: Extraction of bundle-specific structural metrics from crossing fiber models. *NeuroImage*. 2014;100:176-191. doi:10.1016/j.neuroimage.2014.06.015

40. Rheault F, Houde JC, Goyette N, Morency F, Descoteaux M. MI-Brain, a Software to Handle Tractograms and Perform Interactive Virtual Dissection. In: Proceedings of the ISMRM Diffusion study group workshop, Lisbon; 2016.

41. St-Onge E, Schilling KG, Rheault F. BundleSeg: A Versatile, Reliable and Reproducible Approach to White Matter Bundle Segmentation. In: Karaman M, Mito R, Powell E, Rheault F, Winzeck S, eds. *Computational Diffusion MRI*. Springer Nature Switzerland; 2023:47-57. doi:10.1007/978-3-031-47292-3_5

42. Garyfallidis E, Côté MA, Rheault F, et al. Recognition of white matter bundles using local and global streamline-based registration and clustering. *NeuroImage*. 2018;170:283-295. doi:10.1016/j.neuroimage.2017.07.015

43. Cousineau M, Jodoin PM, Garyfallidis E, et al. A test-retest study on Parkinson's PPMI dataset yields statistically significant white matter fascicles. *NeuroImage Clin*. 2017;16:222233. doi:10.1016/j.nicl.2017.07.020

44. Array programming with NumPy | Nature. Accessed April 16, 2024. https://www.nature.com/articles/s41586-020-2649-2

45. Côté MA, Garyfallidis E, Larochelle H, Descoteaux M. Cleaning up the mess : tractography outlier removal using hierarchical QuickBundles clustering. In: Proceedings of: International Society of Magnetic Resonance in Medicine (ISMRM); 2015.

46. Harris CR, Millman KJ, Van Der Walt SJ, et al. Array programming with NumPy. *Nature*. 2020;585(7825):357-362.

47. Virtanen P, Gommers R, Oliphant TE, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nat Methods*. 2020;17(3):261-272. doi:10.1038/s41592-019-0686-2

48. Brett M, Markiewicz CJ, Hanke M, et al. nipy/nibabel: 5.3.1. Published online October 15, 2024. doi:10.5281/zenodo.13936989

SUPPLEMENTARY MATERIALS

Supplementary Material

Download: https://apertureneuro.org/article/154022-tractography-analysis-with-the-scilpy-toolbox/attachment/321869.docx