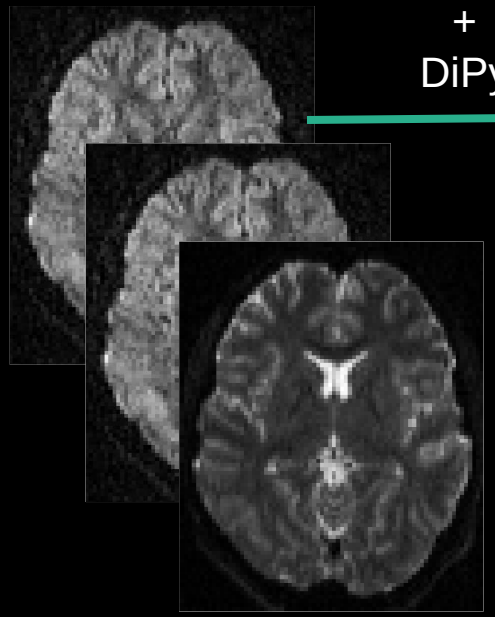
Pipetography:
Diffusion MRI Connectome Generation for HPCs

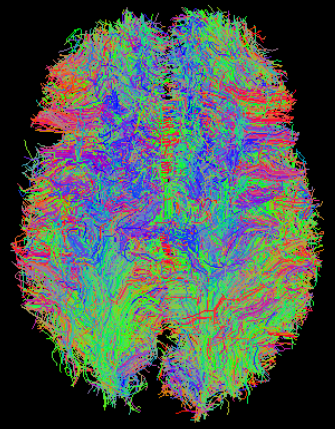
The Brain as a Graph

Diffusion MRI



Pipetography
(FSL)
+
DiPy

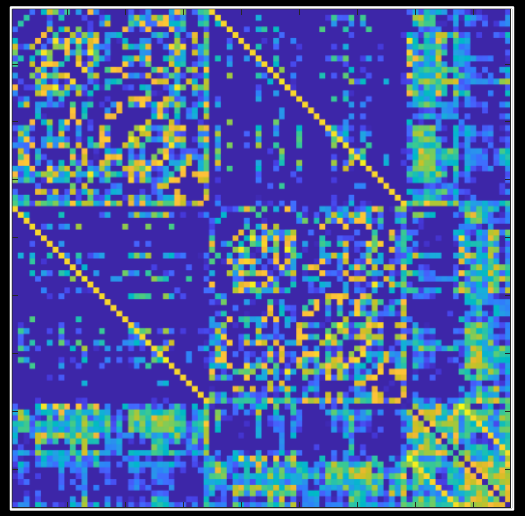
Diffusion Tractography



Pipetography
(MRTrix3 + ANTS)

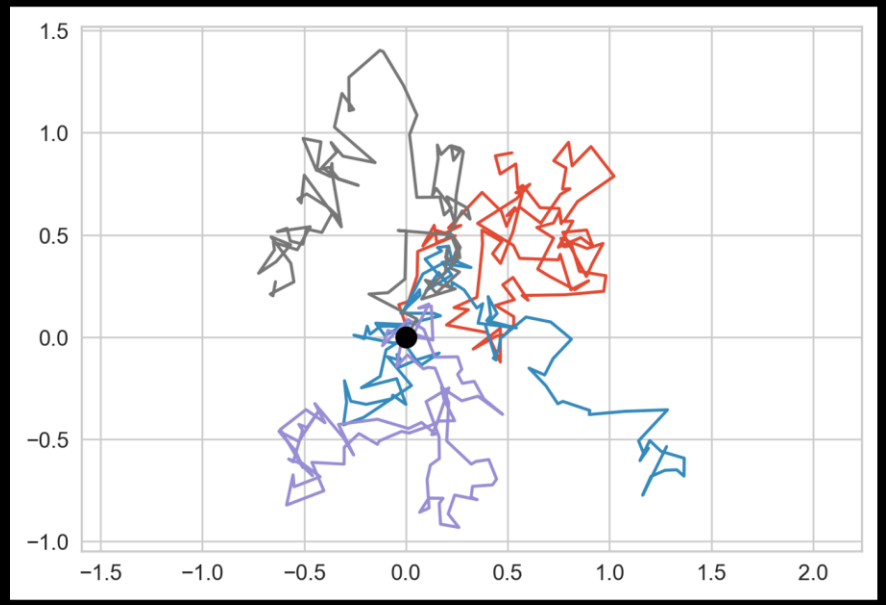
Brain Atlas Defines Nodes

Structural Connectivity

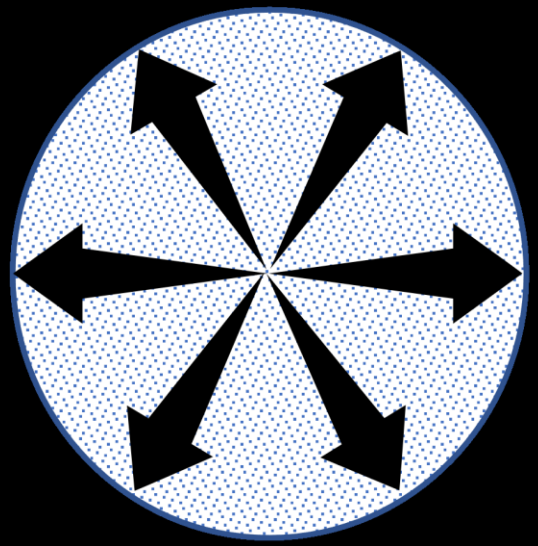


Diffusion MRI Primer: Anisotropy

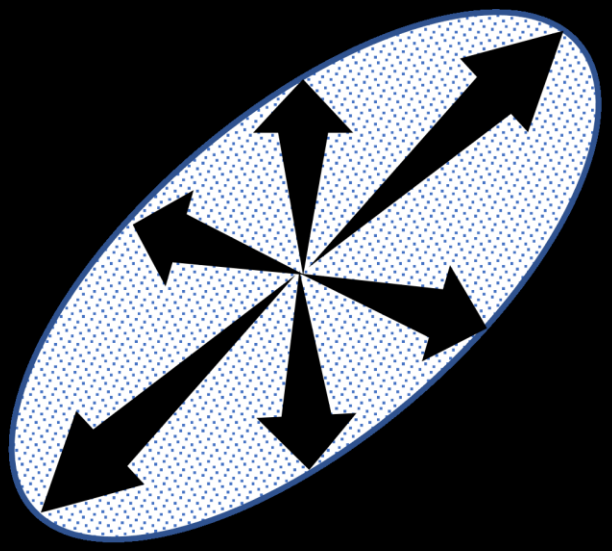
Brownian Motion causes signal decay.



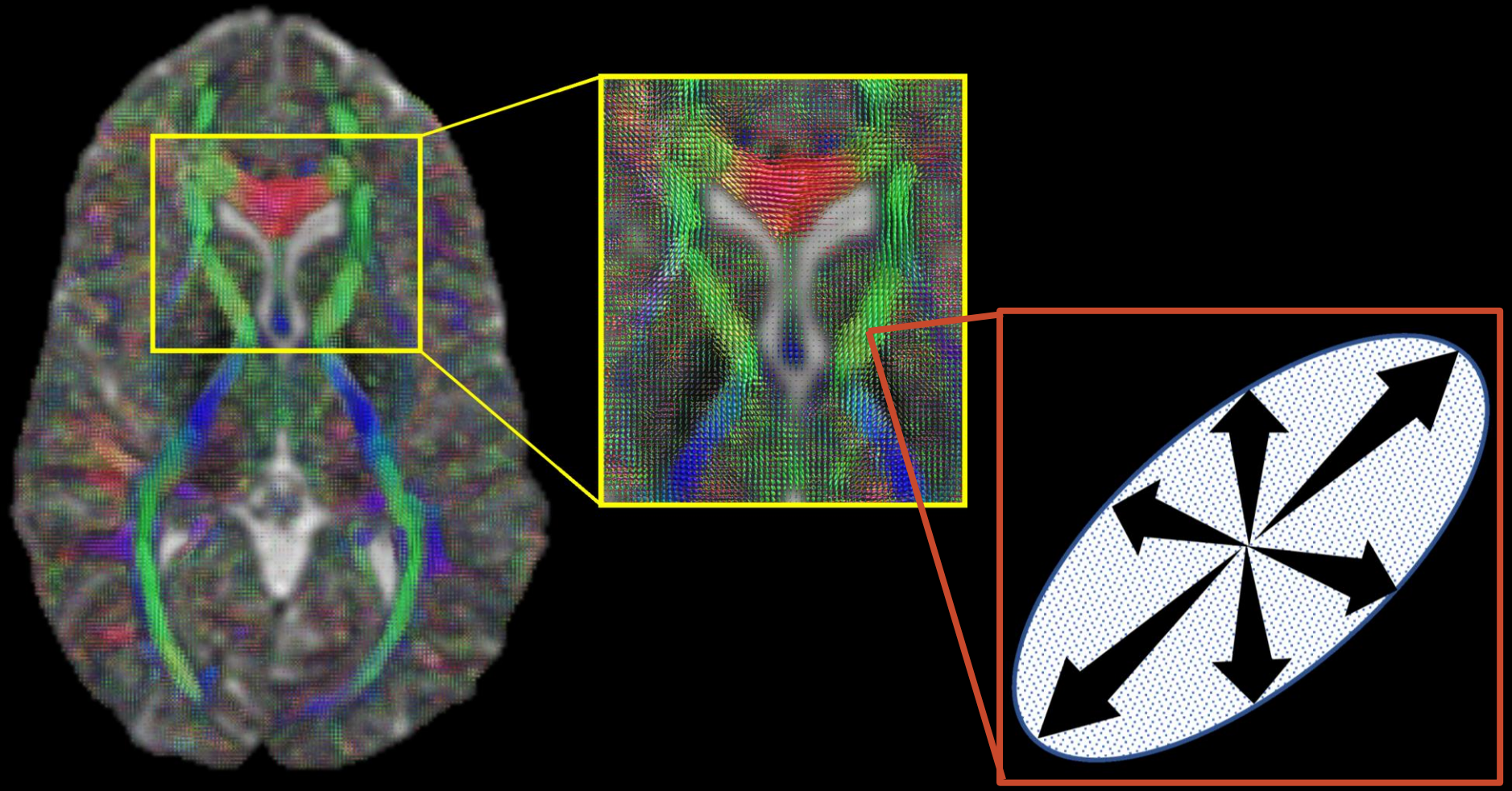
Isotropic Voxel



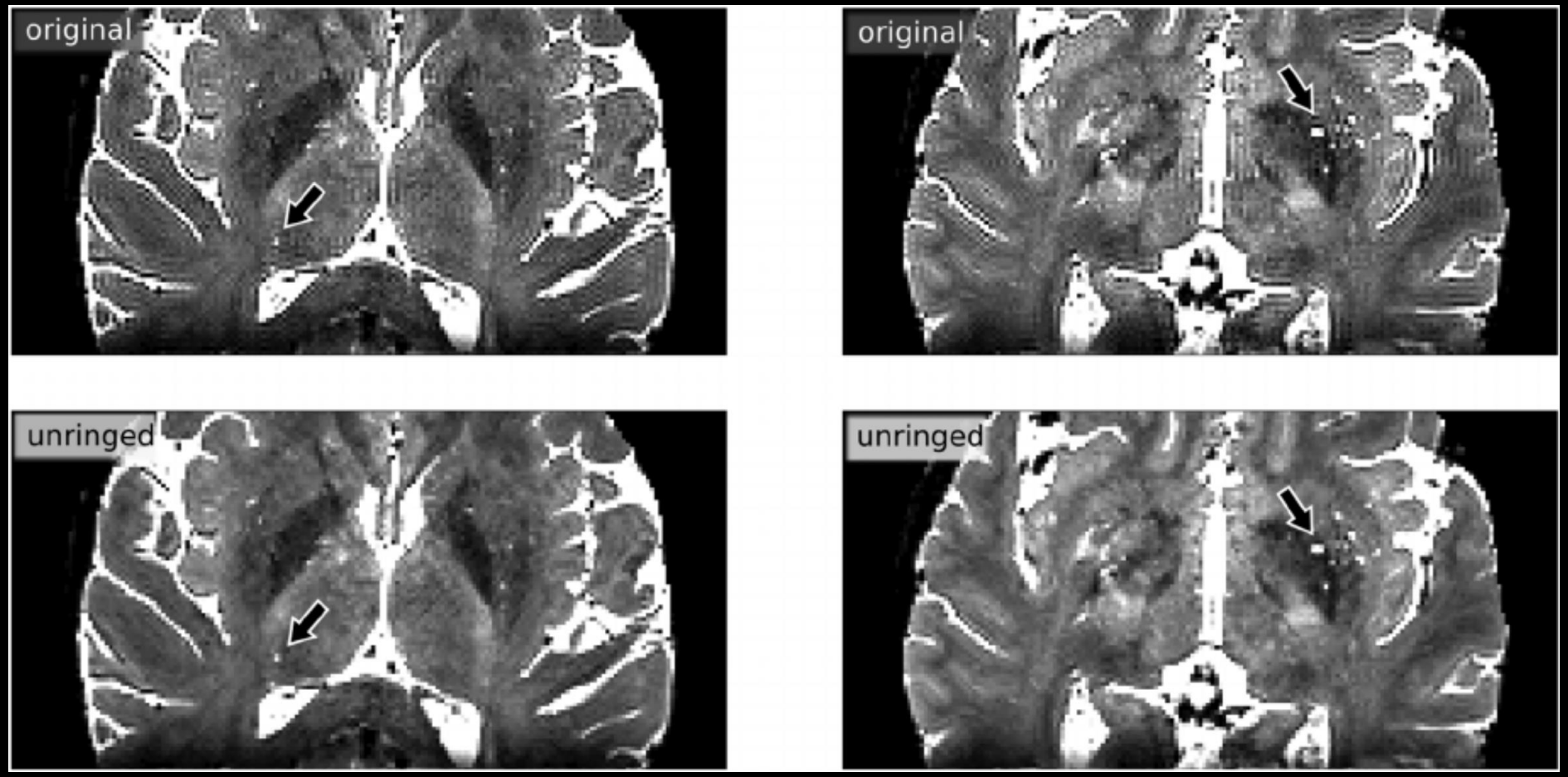
Anisotropic Voxel



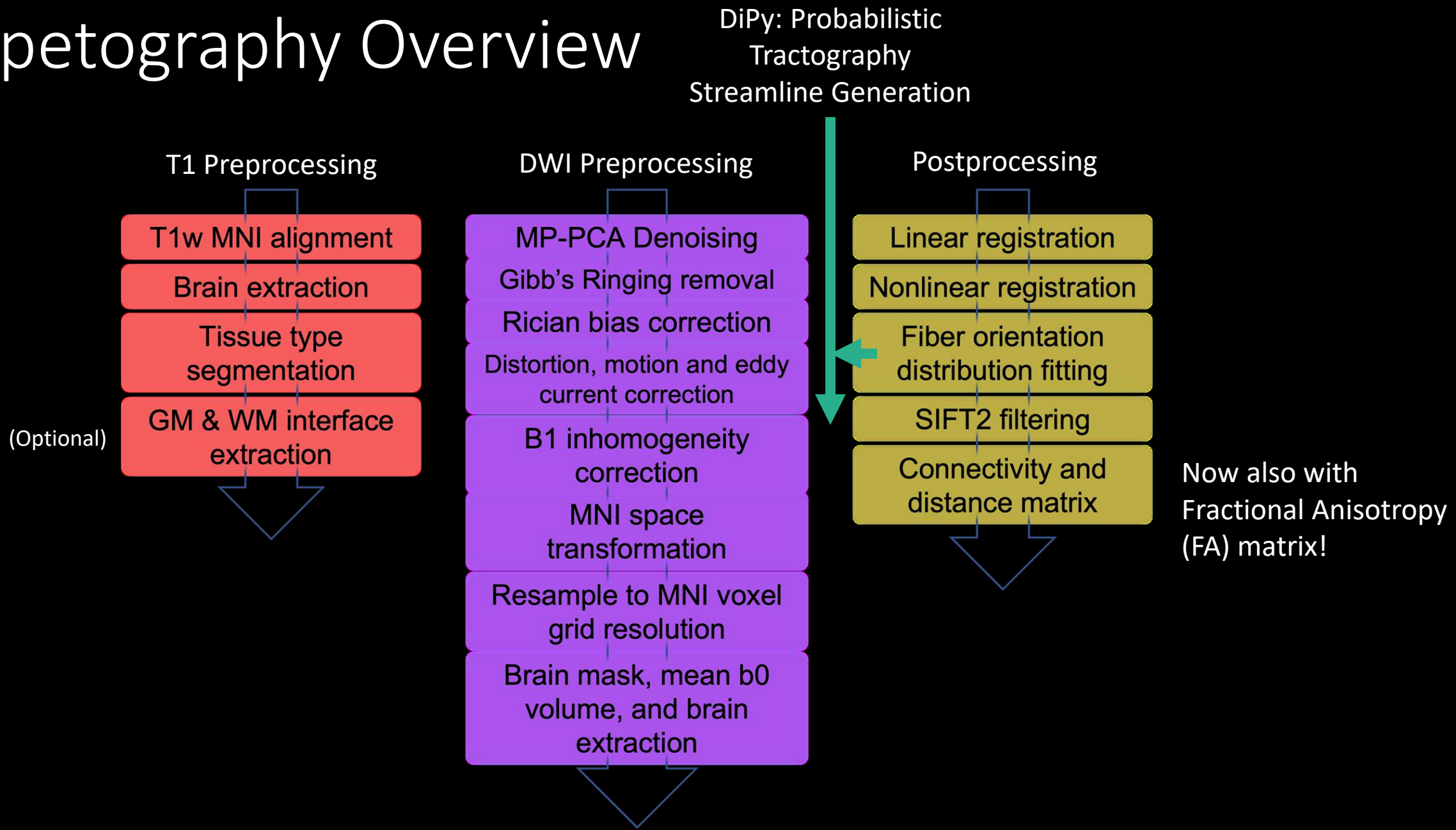
Diffusion MRI Primer: Fiber Orientation Distribution



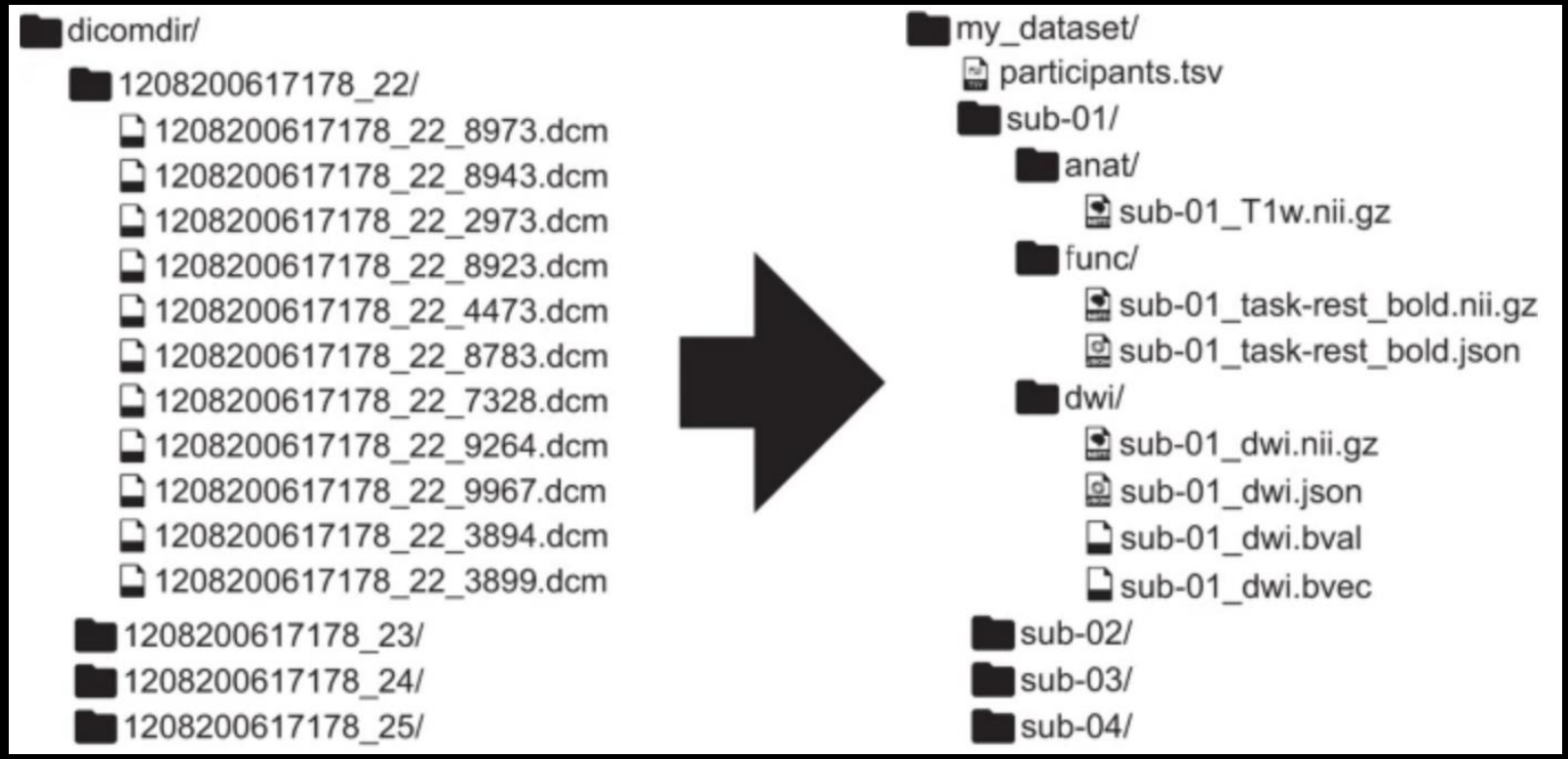
Preprocessing: Removing Artifacts



Pipetography Overview



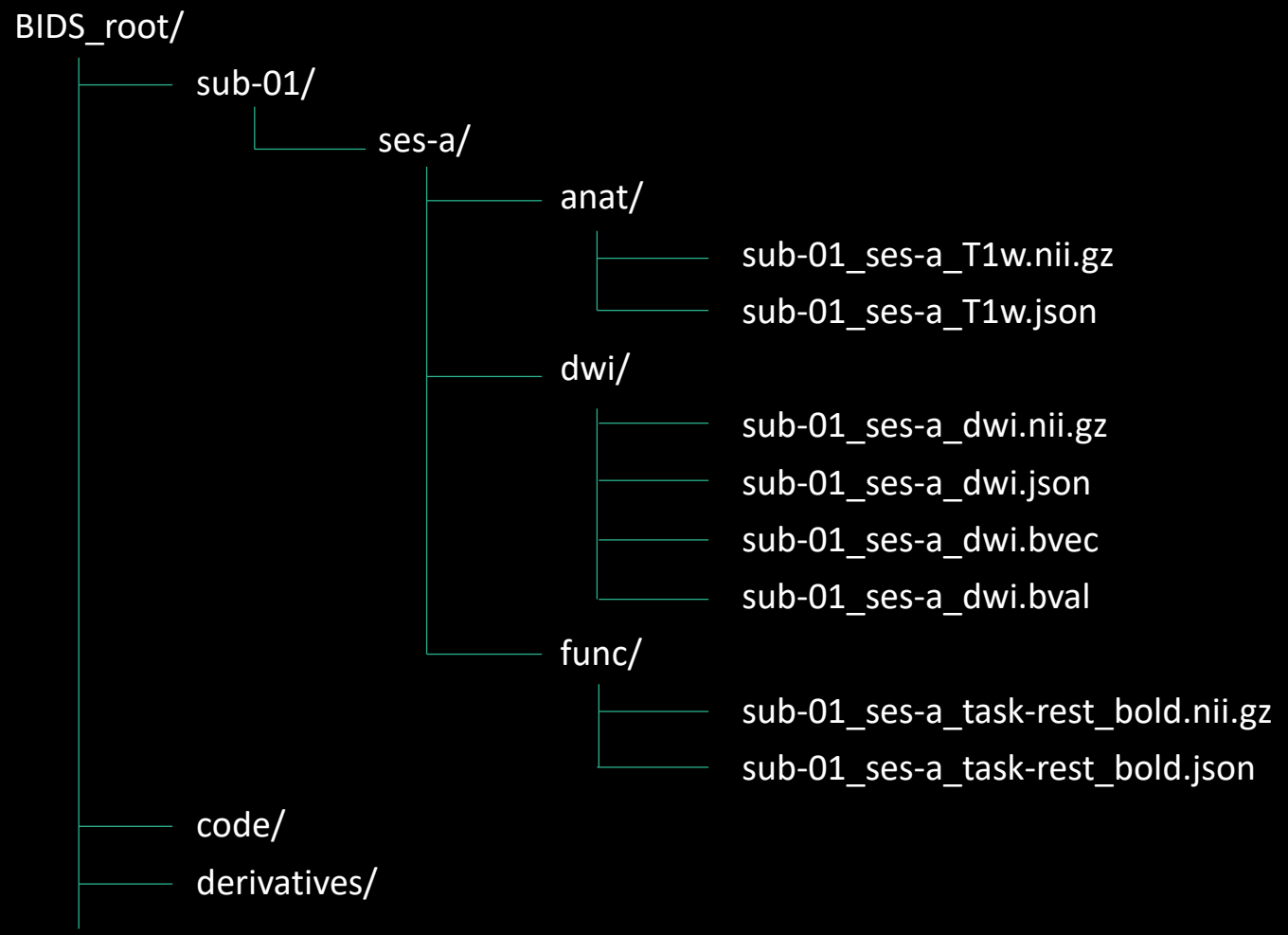
BIDS: Brain Imaging Data Structure



All subjects must be identified by a subject and session ID: sub-**$SUBID**_ses-**$SESID**
Note: may also have a “run” ID.
These IDs are what you will specify to run a pipetography job.

https://bids.neuroimaging.io/

BIDS: Brain Imaging Data Structure



Containerization

- A container is a way to create a frozen (and thus fully reproducible) data processing pipeline.
- A container is basically a file (called an “image”) that is also a self-contained computer.
  - It has installed all the necessary programs (at a specific version) for your analysis.
  - You can execute the same singularity container as many times as you like.
  - ‘.sif’ stands for ‘singularity image format’
- The pipetography container is already on Radiology SCS for use at: /data/i2/software/singularity/pipetography_latest.sif

https://www.freecodecamp.org/news/a-beginner-friendly-introduction-to-containers-vms-and-docker-79a9e3e119b/

Containerization

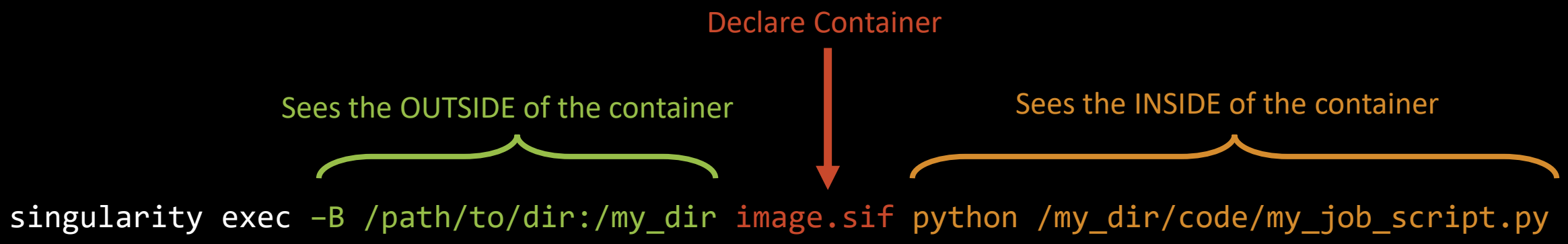
- Three main options to initialize a container:
  - **`shell`**
    - When you want to work with command line inside the container
    - `$> singularity shell image.sif`
  - **`exec`**
    - When you want to give the container a command and script to run
    - `$> singularity exec image.sif python my_job_script.py`
  - **`run`**
    - When the container comes pre-built with a standard default job.
    - This doesn’t apply for pipetography processing
    - `$> singularity run image.sif`

Containers: Binding

- You will probably want to access files that do not exist inside the container, thus you need to bind them.
- To bind files from outside the container, you can use the `-B` flag *before* you call the container.

- The directory outside vs inside do NOT need to be the same in most cases.
- However, some containers need you to bind a directory to a specific location inside the container for output generation.

Containers: Before vs After calling the Image



There are TWO containers for Diffusion Processing!

- /data/i2/software/singularity/pipetography_latest.sif
  - I know this one well, and I have worked on the internal code.
  - This container does dMRI preprocessing and connectome generation.

- /data/i2/software/singularity/gpustreamliens.sif
  - I hardly know anything about this one, and I don't have any way to change its code.
  - This container generates streamlines with a GPU.

(pipetography_latest.sif) (gpustreamlines.sif) (pipetography_latest.sif)

- Preprocessing -> Streamlines -> Connectome
- If using Wynton, you will need to copy over these containers to a local or mounted directory you can access.

SLURM and SGE job submissions

```
#!/bin/bash
# Wynton Submissions:
#$ -S /bin/bash
#$ -N PP_Connect
#$ -cwd
#$ -j y
#$ -l mem_free=40G
#$ -l h_rt=12:00:00 #note: time limit is 7-days
#$ -o /wynton/protected/home/rad-brain/bsipes/outlog_general
```

```
# Radiology SCS Submissions:
#SBATCH --job-name=PP_Connect
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=10
#SBATCH --mem=40G
#SBATCH --time=12:00:00 #note: time limit is 2-days
#SBATCH --output=/home/%u/slurm/%x-%j.log
```

SLURM and SGE job submissions

1. Be sure your data is in BIDS format

2. Be sure all the necessary scripts are in the `/BIDS_root/code` directory on your favorite cluster.

3. Go to `/BIDS_root/code` and submit the job
    1. **SGE**: `qsub preprocess_job_singlesub.sh $sub $ses`
    2. **SLURM**: `sbatch preprocess_job_singlesub.sh $sub $ses`

Personalizing your submission

- Note that to run in maximum parallel, you need to specify the subject and session IDs you want to process.
  - It’s possible to run all subjects/sessions in one job; this is the default behavior for pipetography. It’s much less time efficient but may be easier for small datasets.
- BIDS does not demand any specific convention for naming subjects or sessions, so…
- For the pipetography definition file to recognize your specific subject & session, you may need to do some customization.
  - E.g. do your subject IDs have number or letters or both? Will the variable you pass need to be an integer or a string?

Some possible errors…

- `nx.NetowrkXError("Graph has no nodes or edges")`
  - This means that no subjects were recognized by your sub & ses declaration (i.e. all subjects were excluded from the analysis)
- `tcksift2: [ERROR] invalid first line for key/value file "…" (expected "mrtrix tracks")`
  - This error is relatively rare, has no clear fix, and sometimes goes away after re-running the subject’s streamline generation.

Effective Debuging Procedure

1. If intermediate files were deleted… then in the Python definition file, when constructing the pipetography processing object, set `debug = True`
2. If intermediate files were kept, find them in the working directory. Find the correct subject/session folder, go to the folder with the step that had the error, and find the `command.txt` file.
3. Run singularity as a `shell` and execute the command in the above `command.txt` file to see exactly what's happening. Usually this illuminates the problem.

Note on Streamline Generation

- It may be ideal to use the Gray-Matter-White-Matter Interface (GMWMI) to seed for probabilistic streamline generation.
- We have found that the GMWMI generation does not produce good results for all subjects, so instead we do whole-brain seeing for tractography, which is less biased by GMWMI performance.
- Ultimately, this shouldn’t change the results very much, since the streamline counts are filtered with SIFT-2 and are quantified into the connectivity matrix by GM intersections with an atlas.

Demonstration?

https://github.com/benjaminsipes/Pipetography-HPCCode