# DIPY

## From Diffusion Data to Bundle Analysis

**Gabriel Girard**
gabriel.girard@epfl.ch

**Computational Brain Connectivity Mapping**
**Juan-les-Pins, France**
**20 November 2017**

# DIPY.org

**Diffusion Imaging in Python**

DIPY is an international project which brings together scientists **across labs and countries …**

… to share their state-of-the-art code and expertise in the same codebase, accelerating scientific research in medical imaging.

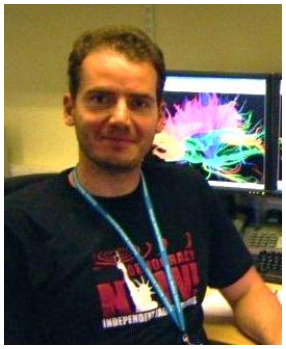
# • Who is dipy for?

**Computer scientists, Engineers, Mathematicians**

- ***Development* and testing of new methods**
- ***Sharing* code and the support/maintenance of that code**
- ***Processing* diffusion MRI data with the state-of-the-art pipeline**
- ***Learning* diffusion MRI processing**

***You need to read and write code***

**Eleftherios Garyfallidis, Ph.D.**
Founder and lead software engineer
elef@indiana.edu



## Some of the Dipy developers

# Development team grows and grows …


Currently the largest diffusion MRI development team
60+ contributors, 10+ countries

# Publications

Citations: 182

METHODS ARTICLE

Front. Neuroinform., 21 February 2014 | https://doi.org/10.3389/fninf.2014.00008

## Dipy, a library for the analysis of diffusion MRI data

Eleftherios Garyfallidis[1,2*], Matthew Brett[3], Bagrat Amirbekian[4], Ariel Rokem[5], Stefan van der Walt[6], Maxime Descoteaux[2], Ian Nimmo-Smith[2] and Dipy Contributors[7]

[1]Computer Science Department, University of Sherbrooke, Sherbrooke, QC, Canada
[2]MRC Cognition and Brain Sciences Unit, University of Cambridge, Cambridge, UK
[3]Henry H. Wheeler, Jr. Brain Imaging Center, University of California, Berkeley, CA, USA
[4]Department of Neurology and Graduate Group in Bioengineering, University of California, San Francisco, CA, USA
[5]Department of Psychology, Stanford University, Stanford, CA, USA
[6]Department of Mathematical Sciences, Division of Applied Mathematics, Stellenbosch University, Stellenbosch, South Africa
[7]http://dipy.org/developers.html

# Industrial use

**Main website:**
dipy.org

**Interactive chat room:**
gitter.im/nipy/dipy/

**Source code:**
github.com/nipy/dipy

**Tutorials:**
dipy.org/examples_index.html

**Dipy Installation:**
nipy.org/dipy/installation.html

**Follow the instructions for your platform.**

```
>> pip install dipy
```

- **Local Reconstruction**
  - Diffusion Tensor
  - Constrained Spherical Deconvolution
- **Streamline Tractography**
  - Masking strategies
  - Propagation algorithms
- **Streamline Bundle Analysis**
  - QuickBundles
  - Regions-based

- **Loading DW-MRI data**

```
fdwi = 'HARDI150.nii.gz'
fbval = 'HARDI150.bval'
fbvec = 'HARDI150.bvec'
```

```
import nibabel as nib
img = nib.load(fdwi)
data = img.get_data()
```

Load data

```
In [50]: print(data.shape)
(81, 106, 76, 160)
```

```
from dipy.io import read_bvals_bvecs
bvals, bvecs = read_bvals_bvecs(fbval, fbvec)
from dipy.core.gradients import gradient_table
gtab = gradient_table(bvals, bvecs)
```

Create the gradient table

```
from dipy.data import fetch_stanford_hardi, read_stanford_hardi
fetch_stanford_hardi()
img, gtab = read_stanford_hardi()
data = img.get_data()
```

Get the data from Dipy

- **Local Reconstruction - DTI**

```
from dipy.segment.mask import median_otsu
maskdata, mask = median_otsu(data, 3, 1, dilate=2)
```

Compute a brain mask

```
In [50]: print(data.shape)
(81, 106, 76, 160)
```

```
In [51]: print(mask.shape)
(81, 106, 76)
```

```
In [52]: print(maskdata.shape)
(81, 106, 76, 160)
```

```
import dipy.reconst.dti as dti
tenmodel = dti.TensorModel(gtab)
tenfit = tenmodel.fit(maskdata)
```

DT Model

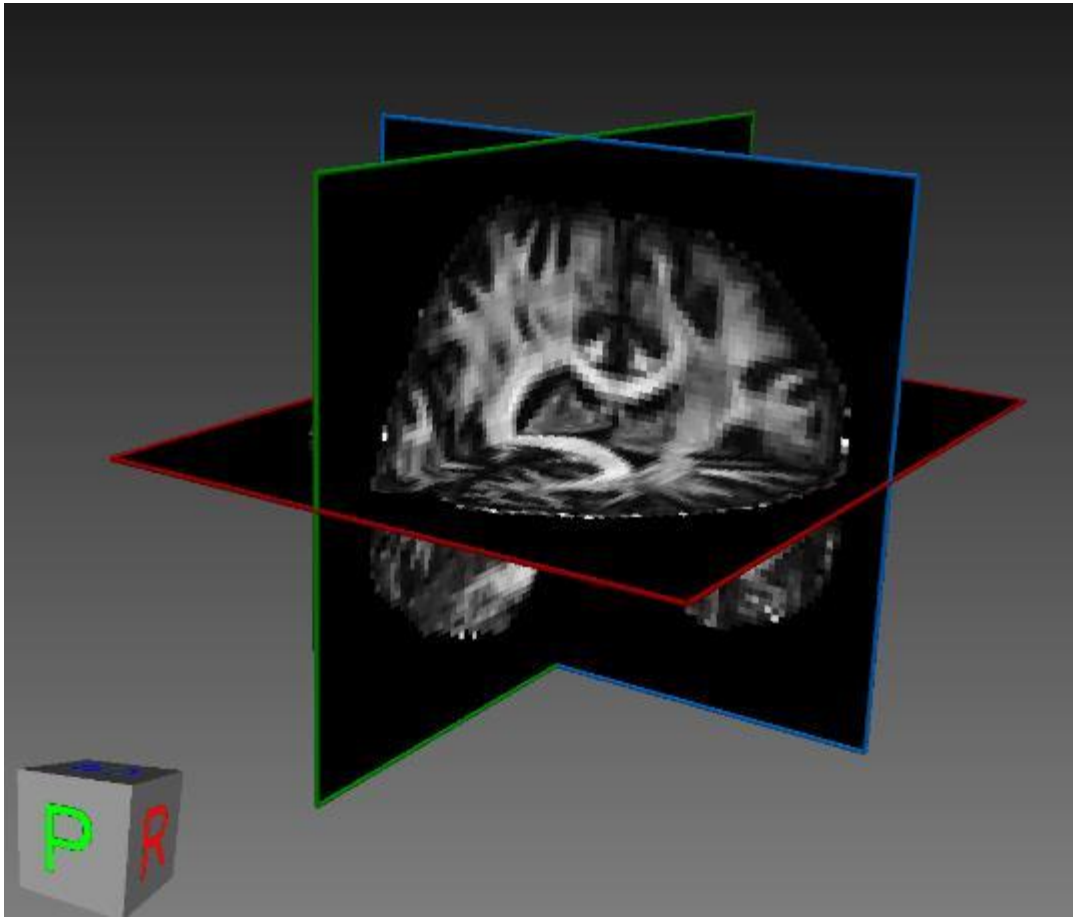
```
FA = fractional_anisotropy(tenfit.evals)
FA[np.isnan(FA)] = 0
```

Compute the FA

```
fa_img = nib.Nifti1Image(FA.astype(np.float32), img.affine)
nib.save(fa_img, 'tensor_fa.nii.gz')
```

Save the FA (Nifti format)

- **Local Reconstruction - DTI**

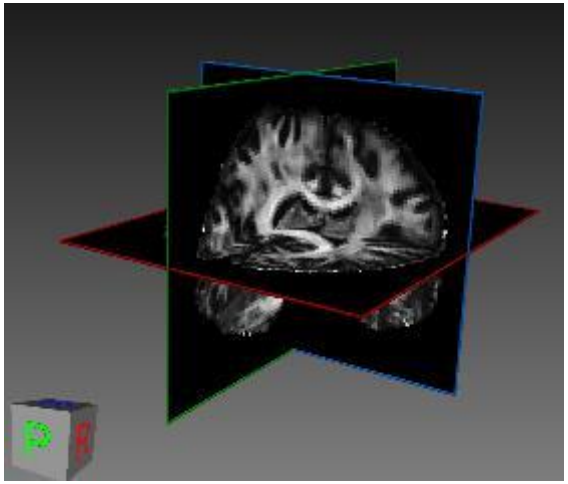



Fractional Anisotropy Map

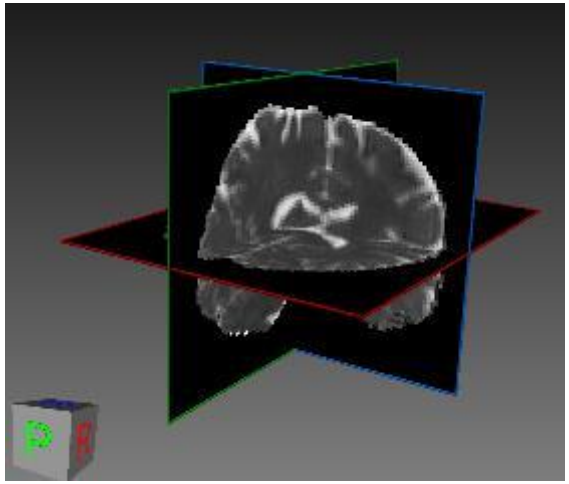
- **Local Reconstruction - DTI**

```
MD = tenfit.md
md_img = nib.Nifti1Image(MD.astype(np.float32), img.affine)
nib.save(md_img, 'tensor_md.nii.gz')
```

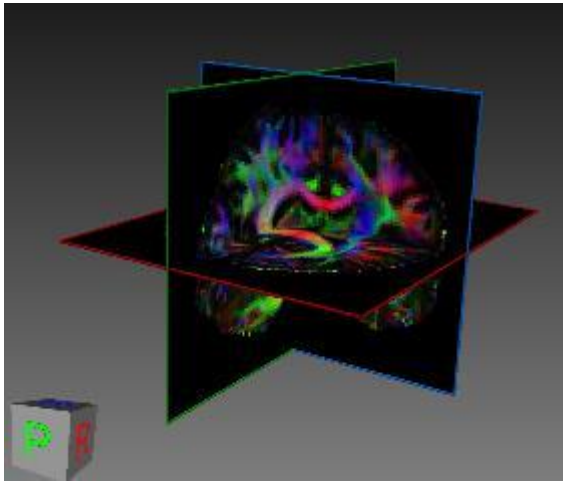
```
FA = np.clip(FA, 0, 1)
RGB = color_fa(FA, tenfit.evecs)
nib.save(nib.Nifti1Image(np.array(255 * RGB, 'uint8'), img.affine), 'tensor_rgb.nii.gz')
```



FA



MD



COLOR-FA

- **Local Reconstruction - DTI**

```
evals = tenfit.evals[28:38, 72:82, 38:39]
evecs = tenfit.evecs[28:38, 72:82, 38:39]
```

Get the tensors evals/evecs

```
cfa = RGB[28:38, 72:82, 38:39]
```

Get the color from the RGB map

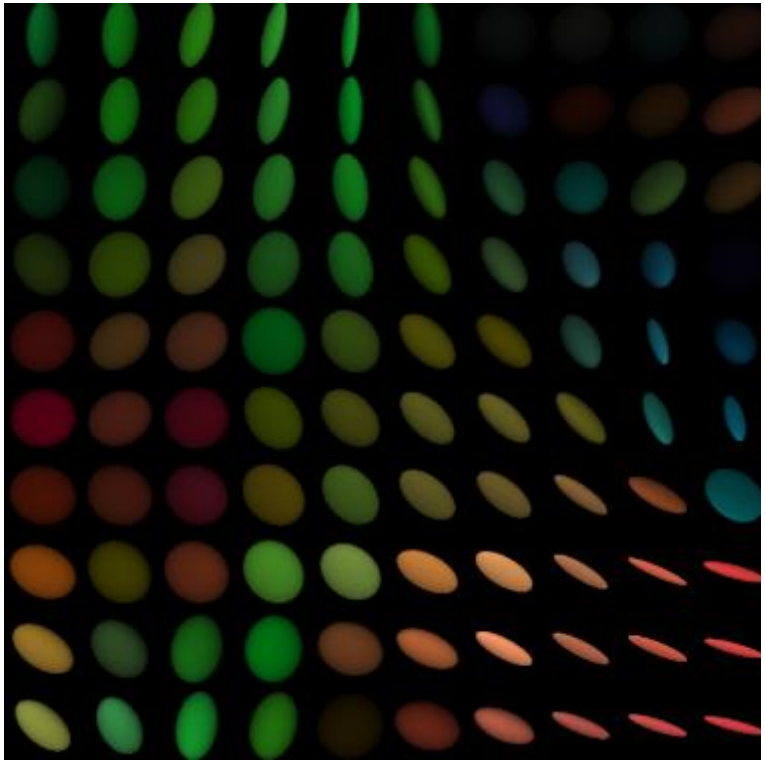
```
from dipy.viz import fvtk
ren = fvtk.ren()
from dipy.data import get_sphere
sphere = get_sphere('symmetric724')
```

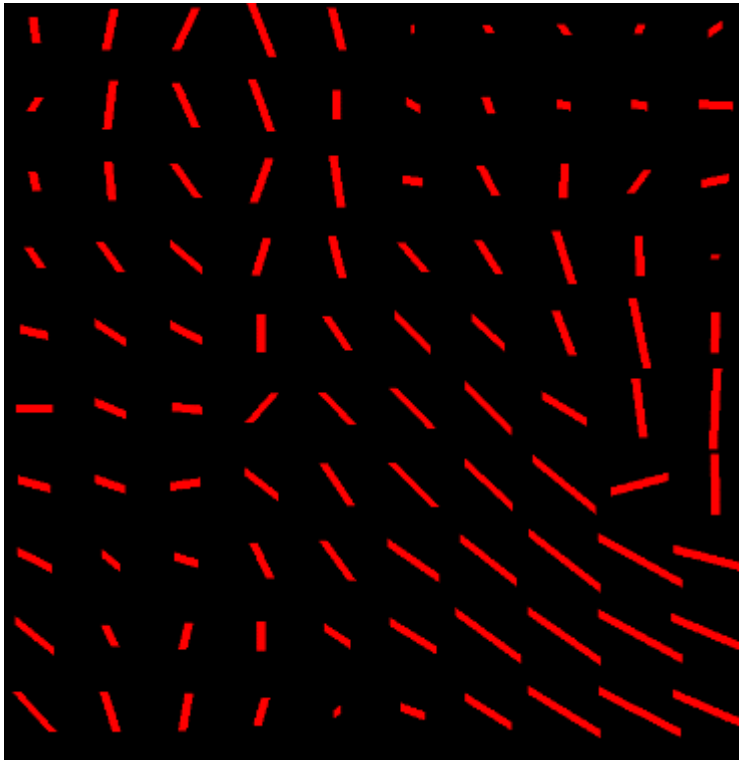
Load the visualization package of Dipy

```
fvtk.add(ren, fvtk.tensor(evals, evecs, cfa, sphere))
fvtk.record(ren, n_frames=1, out_path='tensor_ellipsoids.png', size=(1000, 1000))
```

- **Local Reconstruction - DTI**

```
fvtk.clear(ren)
tensor_peaks = fvtk.peaks(evecs[:,:,0:1,0:1,:], FA[28:38, 72:82, 38:39, None], scale=1.3)
fvtk.add(ren, tensor_peaks)
fvtk.record(ren, out_path='tensor_peaks.png', size=(1000, 1000))
```



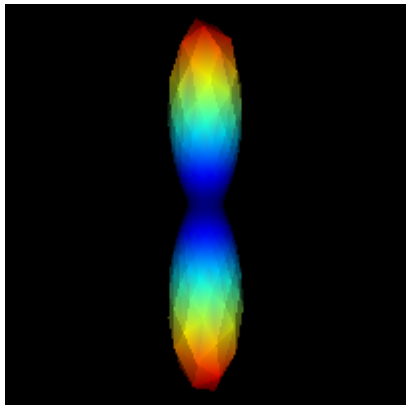


- **Local Reconstruction - CSD**

```
from dipy.reconst.csdeconv import auto_response
response, ratio = auto_response(gtab, data, roi_radius=10, fa_thr=0.7)
```

Estimate the single fiber response

```
from dipy.viz import fvtk
ren = fvtk.ren()
evals = response[0]
evecs = np.array([[0, 1, 0], [0, 0, 1], [1, 0, 0]]).T
from dipy.data import get_sphere
sphere = get_sphere('symmetric724')
from dipy.sims.voxel import single_tensor_odf
response_odf = single_tensor_odf(sphere.vertices, evals, evecs)
response_actor = fvtk.sphere_funcs(response_odf, sphere)
fvtk.add(ren, response_actor)
fvtk.record(ren, out_path='csd_response.png', size=(200, 200))
```



```
from dipy.reconst.csdeconv import ConstrainedSphericalDeconvModel
csd_model = ConstrainedSphericalDeconvModel(gtab, response)
```

CSD Model

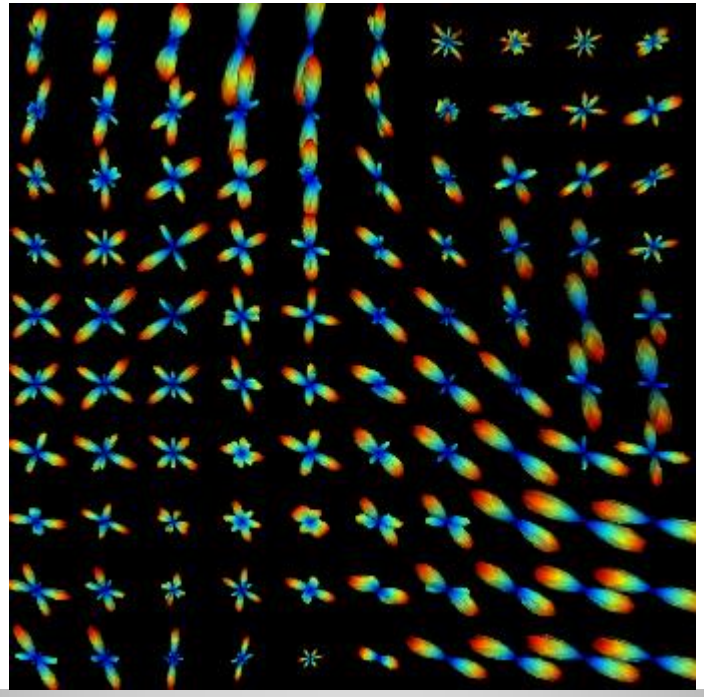
```
csd_fit = csd_model.fit(maskdata)
```

- **Local Reconstruction - CSD**

```
csd_odf = csd_fit[28:38, 72:82, 38:39].odf(sphere)
```

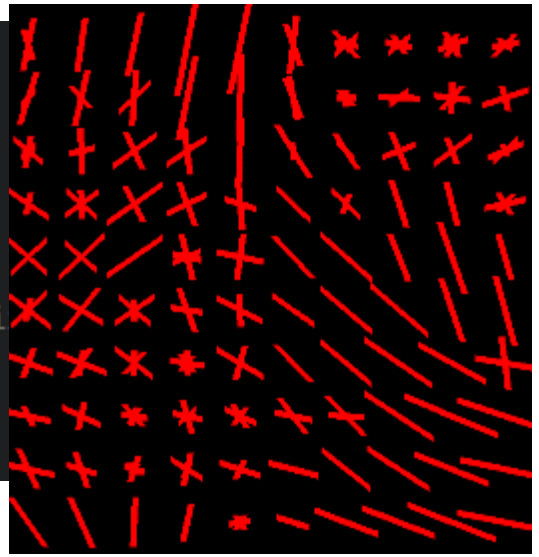
```
In [174]: print(csd_odf.shape)
(10, 10, 1, 724)
```

```
ren = fvtk.ren()
fodf_spheres = fvtk.sphere_funcs(csd_odf, sphere, scale=1.6, norm=False)
fvtk.add(ren, fodf_spheres)
fvtk.record(ren, out_path='csd_odfs.png', size=(1000, 1000))
```



- **Local Reconstruction - CSD**

```
from dipy.direction import peaks_from_model
csd_pfm = peaks_from_model(model=csd_model,
                           data=maskdata,
                           sphere=sphere,
                           return_sh=True,
                           sh_order=8,
                           sh_basis_type='fibernav',  # 'mrtri
                           relative_peak_threshold=.5,
                           min_separation_angle=25,
                           parallel=True)
```



```
csd_peaks_dirs = csd_pfm.peak_dirs[28:38, 72:82, 38:39]
csd_peaks_values = csd_pfm.peak_values[28:38, 72:82, 38:39]
```

```
fvtk.clear(ren)
fodf_peaks = fvtk.peaks(csd_peaks_dirs, csd_peaks_values, scale=1.6)
fvtk.add(ren, fodf_peaks)
fvtk.record(ren, out_path='csd_peaks.png', size=(1000, 1000))
```

```
csd_sh = csd_pfm.shm_coeff
nib.save(nib.Nifti1Image(csd_sh.astype(np.float32), img.affine), 'csd_sh.nii.gz')
```

- Local Reconstruction
  - Diffusion Tensor
  - Constrained Spherical Deconvolution
- **Streamline Tractography**
  - **Masking strategies**
  - **Propagation algorithms**
- Streamline Bundle Analysis
  - QuickBundles
  - Regions-based

- **Tractography Masks – TissueClassifier**
  - **Label Image**
    - ***Binary Tissue Classifier***
  - **Scalar Image (e.g. FA)**
    - ***Threshold Tissue Classifier***
  - **T1 Partial Volume Estimation Maps**
    - ***ACT Tissue Classifier*** [Smith et al., 2012]
    - ***CMC Tissue Classifier*** [Girard et al., 2014]

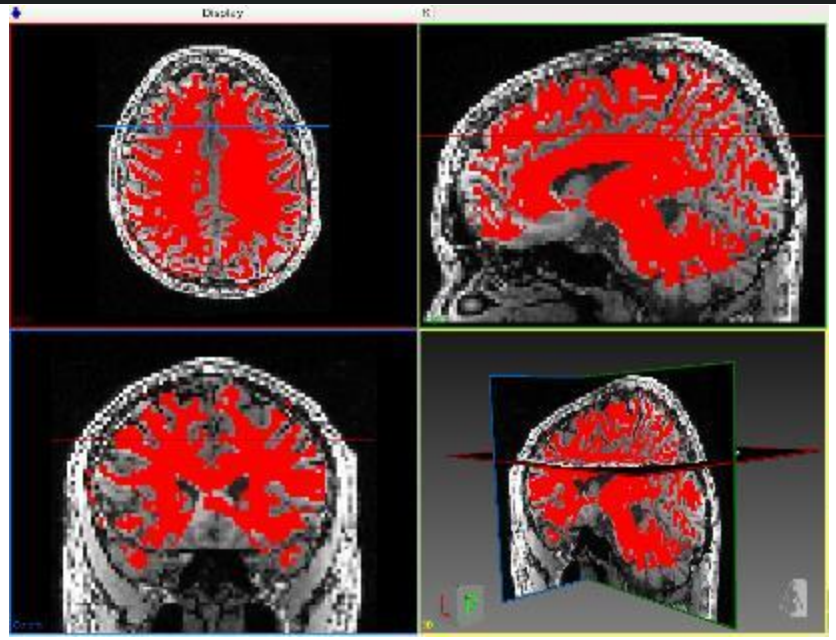
Dipy 0.14.0

# Tractography - Mask

```
from dipy.tracking.local import ThresholdTissueClassifier
threshold_classifier = ThresholdTissueClassifier(FA, .15)
```

Threshold Tissue Classifier

```
mask_fa = FA.copy()
mask_fa[mask_fa < 0.15] = 0
mask_fa[mask_fa>0]=1
nib.Nifti1Image(mask_fa.astype("uint8"),affine).to_filename("mask_fa.nii.gz")
```



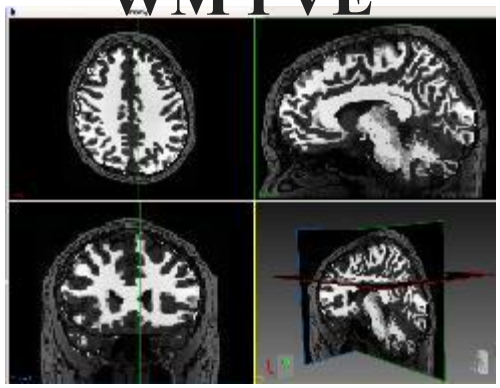
# • Tractography – Mask from T1 Information

```
f_pve_wm='pve_wm.nii.gz'
f_pve_gm='pve_gm.nii.gz'
f_pve_csf='pve_csf.nii.gz'
```

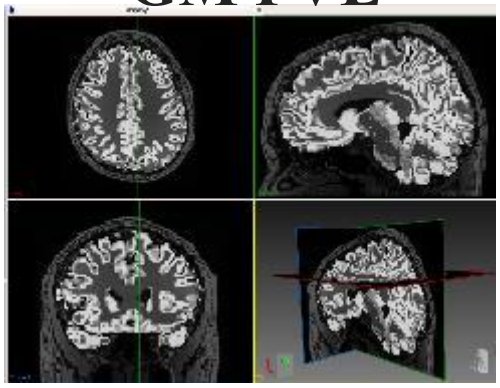
```
img_pve_wm = nib.load(f_pve_wm)
img_pve_gm = nib.load(f_pve_gm)
img_pve_csf = nib.load(f_pve_csf)
```

Load PVE maps
e.g. from FSL-Fast or Dipy

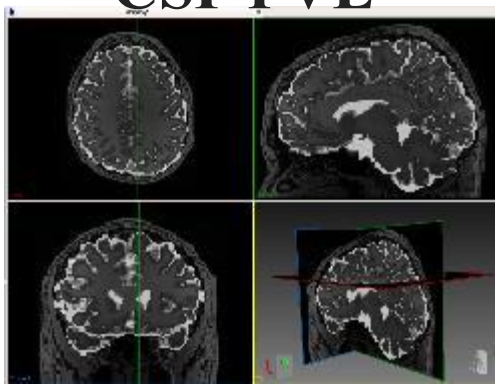
WM PVE



GM PVE



CSF PVE



```
from dipy.tracking.local import ActTissueClassifier
act_classifier = ActTissueClassifier.from_pve(img_pve_wm.get_data(),
                                              img_pve_gm.get_data(),
                                              img_pve_csf.get_data())
```

ACT
Tissue Classifier

Dipy 0.14.0

[Smith et al., 2012]

- **Tractography - Seeding**

```
from dipy.data import read_stanford_labels
_, _, labels_img = read_stanford_labels()
labels = labels_img.get_data()
```

Load a brain parcellation

```
from dipy.tracking import utils
seed_mask = labels == 2
seeds = utils.seeds_from_mask(seed_mask, density=1, affine=affine)
```

1 seed in each voxel of the CC

**`Seeds`** is a list of 3D initial positions

# Tractography Algorithms – DirectionGetters

- **ProbabilisticDirectionGetter**
  - *Samples a direction from the SF values in a maximum angle θ*
- **MaximumDirectionGetter**
  - *The direction with the maximum SF value in a maximum angle θ*

Dipy 0.14.0

- **ClosestPeakDirectionGetter**
  - *The direction of a peak of the SF the most aligned with the previous direction in a maximum angle*

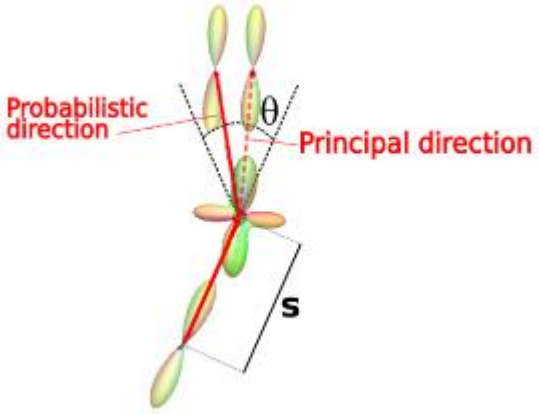
- **EuDx**
  - *Tri-linear interpolation of the closest peaks at the current tracking position*

Dipy 0.14.0

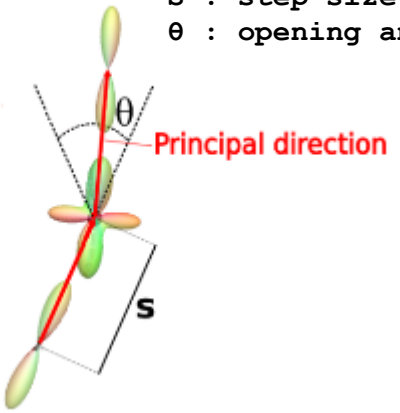
- **BootDirectionGetter**
  - *Sample the direction from the estimated uncertainty in the SF* [Berman et al., 2008]

Dipy 0.14.0

- **Particle Filtering Tractography**
  - **ProbabilisticDirectionGetter with prior anatomical information** [Girard et al., 2014]




S : step size
θ : opening angle




[Cote et al., 2013]

- **Tractography – Direction Getters**

```
from dipy.direction import ProbabilisticDirectionGetter
prob_dg = ProbabilisticDirectionGetter.from_shcoeff(csd_fit.shm_coeff,
                                                    max_angle=20.,
                                                    sphere=sphere)
```

Probabilistic Direction Getter from shm

```
from dipy.direction import DeterministicMaximumDirectionGetter
pmf = nib.load("my_FOD.nii.gz").get_data()
detmax_dg = DeterministicMaximumDirectionGetter.from_pmf(pmf,
                                                         max_angle=45.,
                                                         sphere=sphere)
```

Deterministic Maximum Direction Getter from FOD sampled on a sphere

```
from dipy.direction import ClosestPeakDirectionGetter
shm_coeff = nib.load("sh_FOD.nii.gz").get_data()
closest_dg = ClosestPeakDirectionGetter.from_shcoeff(shm_coeff,
                                                     max_angle=60.,
                                                     sphere=sphere,
                                                     basis="mrtrix")
```

Basis:
[Descoteaux et al., 2007]
[Tounier et al., 2012]

Closest Peak Direction Getter from shm (Nifti)

- **Tractography - Streamlines**

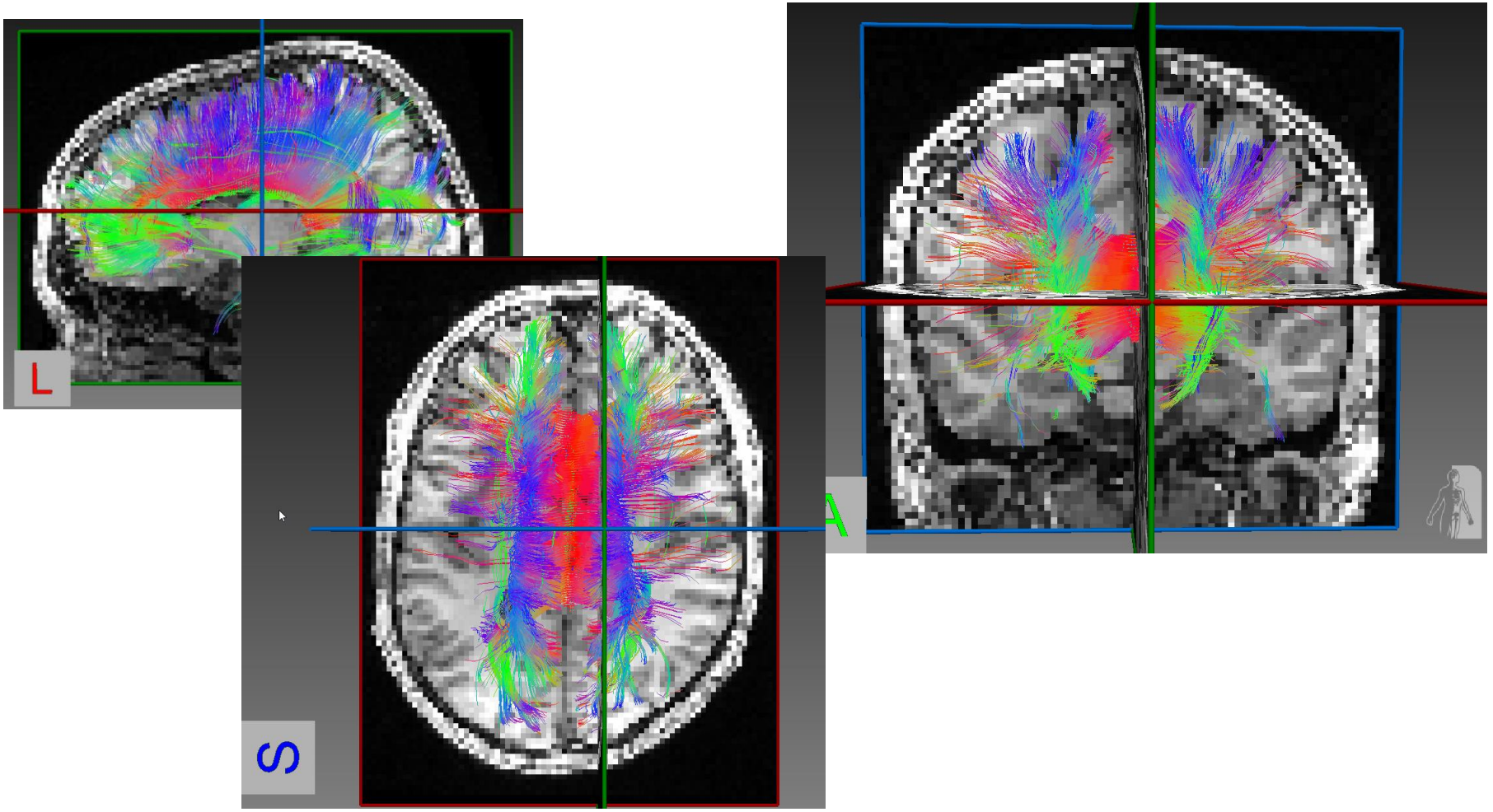
```
from dipy.tracking.local import LocalTracking
streamlines = LocalTracking(prob_dg,
                            act_classifier,
                            seeds,
                            affine,
                            step_size=0.2)
```

Generate streamlines with the selected parameters

```
from dipy.io.trackvis import save_trk
save_trk("probabilistic_streamlines.trk",
         streamlines,
         affine,
         seed_mask.shape)
```

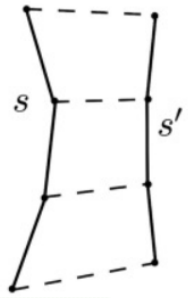
Save the streamlines to the disk

- **Tractography - Streamlines**



- Local Reconstruction
  - Diffusion Tensor
  - Constrained Spherical Deconvolution
- Streamline Tractography
  - Masking strategies
  - Propagation algorithms
- **Streamline Bundle Analysis**
  - **QuickBundles**
  - **Regions-based**

- **Streamline Clustering with QuickBundle**



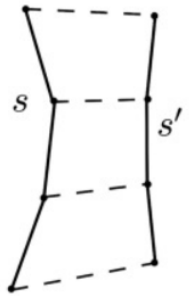

QuickBundles for tractography simplification
Garyfallidis et al. Frontiers 2012

# Streamline Clustering with QuickBundles

```
from dipy.segment.clustering import QuickBundles
qb = QuickBundles(threshold=10.)

clusters = qb.cluster(streamlines)
```

Apply the QuickBundles algorithm to the streamlines




```
threshold : float
    The maximum distance from a bundle for a streamline to be still
    considered as part of it.
```

```
print("Nb. clusters:", len(clusters))
print("Cluster sizes:", map(len, clusters))
```

```
Nb. clusters: 4

Cluster sizes: [64, 191, 47, 1]
```

[Garyfallidis et al., 2012]

- **Streamline Clustering with QuickBundles**

```
print("Streamlines indices of the first cluster:\n", clusters[0].indices)
```

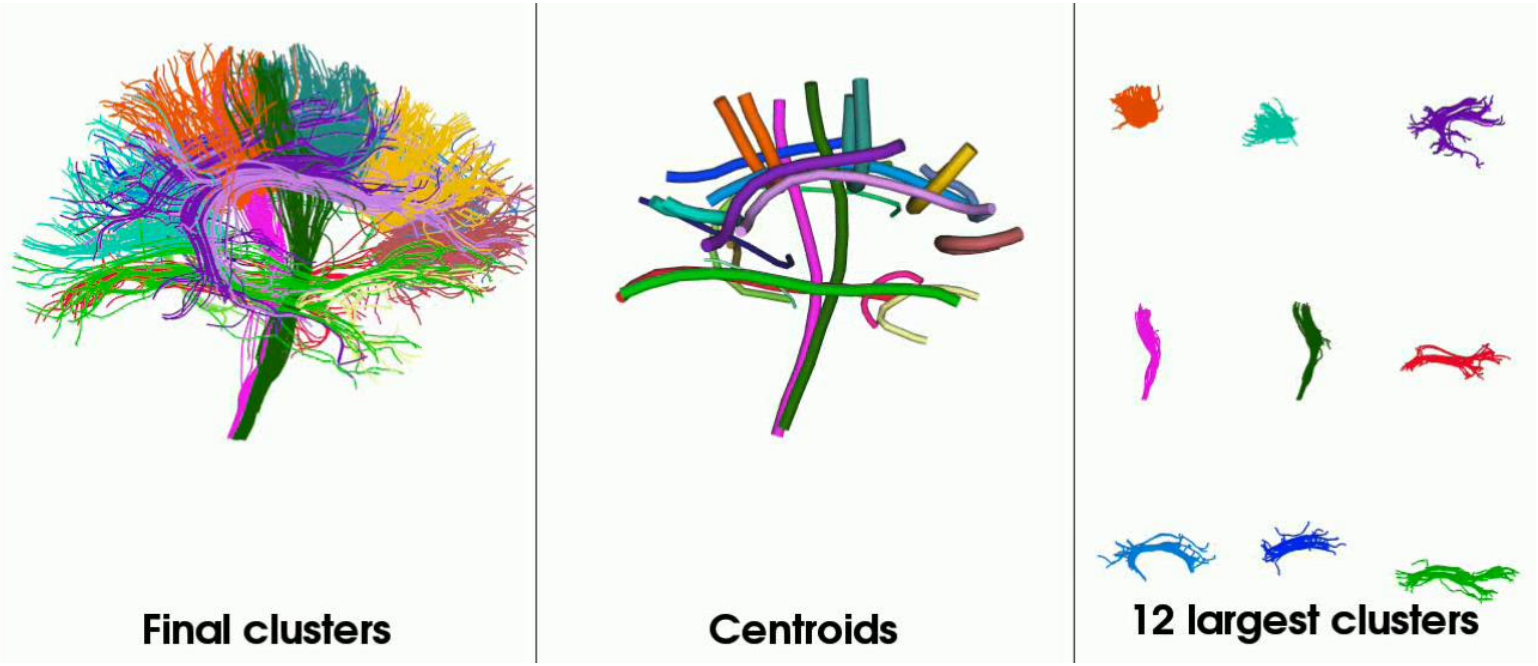
```
Streamlines indices of the first cluster:
[0, 7, 8, 10, 11, 12, 13, 14, 15, 18, 26, 30, 33, 35, 41, 65, 66, 85, 100,
 101, 105, 115, 116, 119, 122, 123, 124, 125, 126, 128, 129, 135, 139, 142,
 143, 144, 148, 151, 159, 167, 175, 180, 181, 185, 200, 208, 210, 224, 237,
 246, 249, 251, 256, 267, 270, 280, 284, 293, 296, 297, 299]
```

```
print("Centroid of the last cluster:\n", clusters[-1].centroid)
```

```
Centroid of the last cluster:
array([[  84.83773804,  117.92590332,   77.32278442],
       [  86.10850525,  115.84362793,   81.91885376],
       [  86.40357208,  112.25676727,   85.72930145],
       [  86.48336792,  107.60327911,   88.13782501],
       [  86.23897552,  102.5100708 ,   89.29447174],
       [  85.04563904,   97.46020508,   88.54240417],
       [  82.60240173,   93.14851379,   86.84208679],
       [  78.98937225,   89.57682037,   85.63652039],
       [  74.72344208,   86.60827637,   84.9391861 ],
       [  70.40846252,   85.15874481,   82.4484024 ],
       [  66.74534607,   86.00262451,   78.82582092],
       [  64.02451324,   88.43942261,   75.0697403 ]], dtype=float32)
```

[Garyfallidis et al., 2012]

- **Streamline Clustering with QuickBundles**




[Garyfallidis et al., 2012]

- **Label-based bundle segmentation**

```
from dipy.tracking import utils
M, grouping = utils.connectivity_matrix(streamlines,
                                        labels,
                                        affine=affine,
                                        return_mapping=True,
                                        mapping_as_streamlines=True)
```

```
In [108]: M.shape
Out[108]: (89, 89)
```

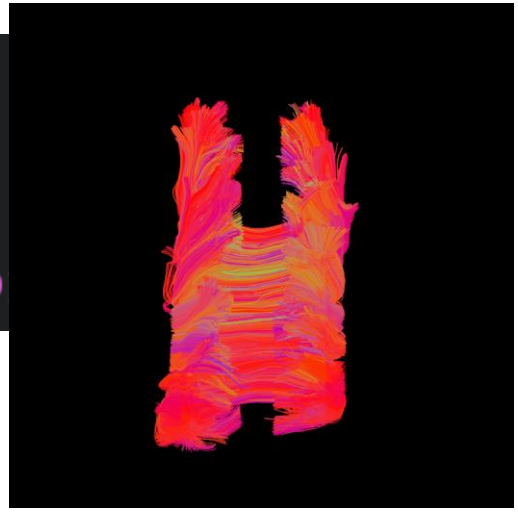
```
In [172]: M[11,54]
Out[172]: 9201
```

```
In [173]: len(grouping[(11,54)])
Out[173]: 9201
```

Superiror frontal gyrus L/R

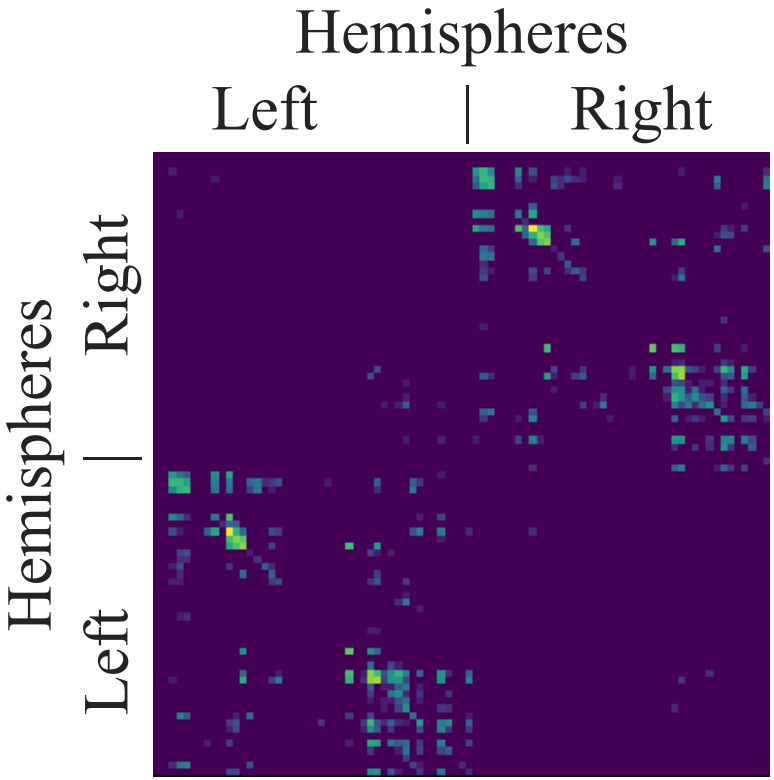
```
bundle = grouping[(11,54)]
streamlines_actor = fvtk.line(bundle, line_colors(bundle))

r = fvtk.ren()
fvtk.add(r, streamlines_actor)
fvtk.record(r, n_frames=1, out_path='bundle.png',  size=(800, 800
```



- **Label-based bundle segmentation**

```
import matplotlib.pyplot as plt
plt.imshow(np.log1p(M), interpolation='nearest')
plt.savefig("connectivity.png")
```



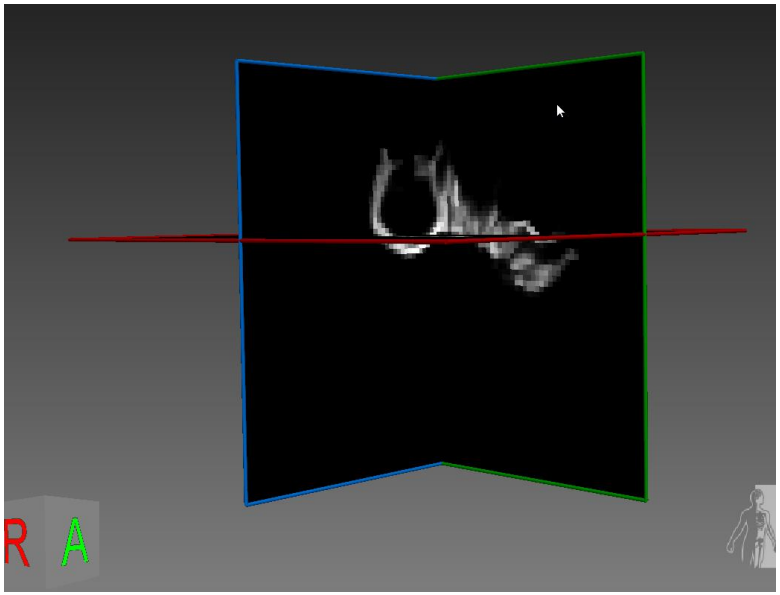

# of Streamlines of
the CC between labels

- **Bundle Density Map**

```
bundle = grouping[(11,54)]
dm = utils.density_map(bundle, shape, affine=affine)
```

Compute streamline density map

```
dm_img = nib.Nifti1Image(dm.astype("int16"), hardi_img.affine)
dm_img.to_filename("bundle_density.nii.gz")
```




# Questions?



- **Preprocessing**
  - SNR Estimation
  - Denoising with Non-Local Means
  - Volume Reslicing
- **Local Reconstructions**
  - Constrained Spherical Deconvolution (CSD)
  - Simple Harmonic Oscillator based Reconstruction and Estimation (3D-SHORE)
  - Mean Apparent Propagator (MAP)-MRI
  - Diffusion Tensor Imaging (DTI)
  - Diffusion Kurtosis Imaging (DTK)
  - Q-Ball Constant Solid Angle (CSA)
  - Diffusion Spectrum Imaging (DSI)
  - Generalized Q-Sampling Imaging (GQI)
- **Registration**
  - Image-based Registration
  - Affine Registration
  - Symmetric Diffeomorphic Registration
  - Streamline-based Registration
- **Segmentation**
  - Streamline Clustering
  - Tractography Clustering with QuickBundles
  - Brain segmentation with median_otsu
  - T1 Tissue Partial Volume Estimation
- **Fiber Tractography**
  - Deterministic / Probabilistic Tractography
  - Particle Filtering Tractography (PFT)
- **Streamline Analysis**
  - Streamline analysis and connectivity
  - Connectivity Matrices
  - ROI Intersections and Density Maps
  - Streamline length and size reduction
  - Linear fascicle evaluation
  - Fiber to bundle coherence measures
- **Visualization**
  - Local Reconstructions
  - Streamlines

**See the code examples: Dipy.org**

**Thank you for your attention!**