NiPype: A Pypelined Approach to Neuroimaging Data Analysis

http://nipy.sourceforge.net/nipype

Satrajit Ghosh satra@mit.edu

Massachusetts Institute of Technology

Acknowledgements

People

core nipype developers:
chris burns, dav clark, satrajit ghosh, cindee madison

other contributors (financial, social, technical and testing support):
matthew brett, mark d'esposito, stefan ehrlich, the gablab, john gabrieli, randy gollub, scott gorlin, chris gorgolewski, doug greve, yaroslav halchenko, michael hanke, oliver hinds, arno klein, jarrod millman, alfonso nieto-castanon, mark pearrow, fernando perez, jean baptiste-poline, alexis roche, yannick schwartz, jonathon taylor, jason tourville, bertrand thirion, rosalia tungaraza, gael varoquaux, susan whitfield-gabrieli ...

my closest supporters:
amie, katrien

Funding

NIBIB
R03 EB008673
PIs: Satrajit Ghosh, Susan Whitfield-Gabrieli, MIT

NIMH
R01 MH081909
PIs: Marc D'Esposito, UC, Berkeley

Software

FSL
FreeSurfer
NiPy
SPM

Institutions

Boston University
Dartmouth College
Massachusetts Institute of Technology
Neurospin, France
Stanford
University of California, Berkeley
University of Edinburgh
University of Washington, Seattle

origin

python

goals

architecture

challenges

Neuroimaging Pypelines	origin

Region-of-interest Analysis of Parcellated Imaging Data

Reducing inter-subject anatomical variability

ARtifact detection Tools

Quality assurance

Neuroimaging Pypelines	origin

Region-of-interest Analysis of Parcellated Imaging Data
Reducing inter-subject anatomical variability

ARtifact detection Tools
Quality assurance

MATLAB-based
SPM-focused
Lab-centric

Neuroimaging Pypelines | distribution

solutions?

solutions?

- Write it in C/C++

solutions?

- Write it in C/C++
- Wrap it with swig/mex

solutions?

- Write it in C/C++
- Wrap it with swig/mex
- Distribute with MatlabRuntime

solutions?

- Write it in C/C++
- Wrap it with swig/mex
- Distribute with MatlabRuntime

or more problems?

solutions?

- Write it in C/C++
- Wrap it with swig/mex
- Distribute with MatlabRuntime

or more problems?

- Code maintenance

solutions?

- Write it in C/C++
- Wrap it with swig/mex
- Distribute with MatlabRuntime

or more problems?

- Code maintenance
- Code readability

solutions?

- Write it in C/C++
- Wrap it with swig/mex
- Distribute with MatlabRuntime

or more problems?

- Code maintenance
- Code readability
- Interoperability

solutions?

- Write it in C/C++
- Wrap it with swig/mex
- Distribute with MatlabRuntime

or more problems?

- Code maintenance
- Code readability
- Interoperability
- Data formats

Neuroimaging Pypelines	data formats

SPM	FSL	FreeSurfer
nii (3D/4D)	nii(.gz)	nii(.gz)
img/hdr (analyze, nifti)		mgh(gz)
MAT (matlab)		

Easy to program and document

Good numerical support

Cross-platform

Easy to distribute

Existing neuroimaging infrastructure
(support for multiple data formats)

Neuroimaging Pypelines

python!

Neuroimaging Pypelines | python

6.00 Introduction to Computer Science and Programming

As taught in: Fall 2008

Many of the problem sets focus on specific topics, such as virus population dynamics, word games, protein sequences, or simulating the movement of a Roomba. (Roomba photograph courtesy of Stephanie Booth on Flickr; virus image courtesy of the CDC; Boggle photograph courtesy of Angelina on Flickr; protein image courtesy of the Lawrence Berkeley National Laboratory.)

Level:
Undergraduate

Instructors:
Prof. Eric Grimson
Prof. John Guttag

Course Features
Course Description
Technical Requirements

Course Features

- Video lectures
- Exams and Solutions
- Assignments (no solutions)

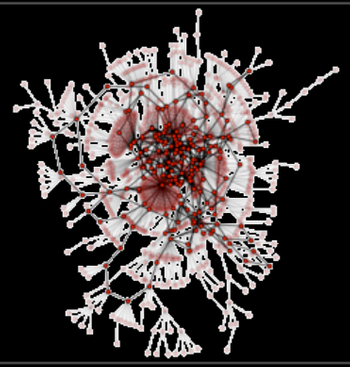
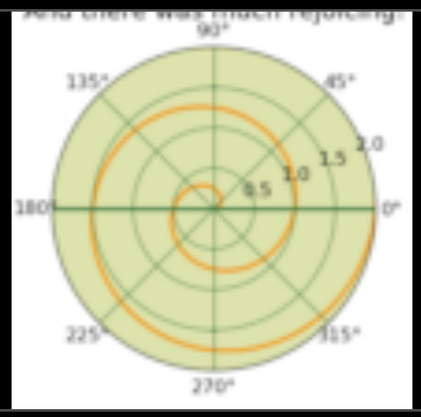
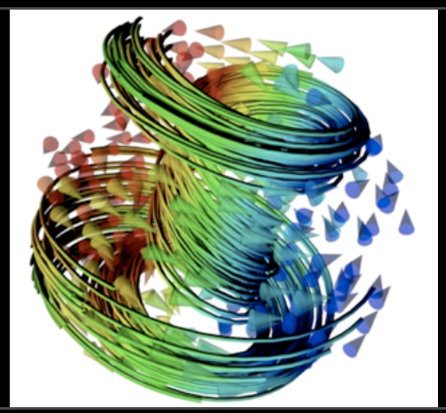
Course Description

This subject is aimed at students with little or no programming experience. It aims to provide students with an understanding of the role computation can play in solving problems. It also aims to help students, regardless of their major, to feel justifiably confident of their ability to write small programs that allow them to accomplish useful goals. The class will use the Python™ programming language.

Neuroimaging Pypelines | python

- High-level language (encourages code reuse)
- Well-designed language
- Cross-platform
- Implementations are freely available
- Extensive community and commercial support
- Strong numerical and scientific computing support
- Interactive command line (instant development)
- Test-Driven Development
- Documentation generation made easy

Neuroimaging Pypelines | scientific python

ipython	scipy + numpy	networkx
IP[y]:		
sympy	matplotlib	mayavi + tvtk
		

sage

python(x,y)

ENTHOUGHT
SCIENTIFIC COMPUTING SOLUTIONS

scipy:
statistics
optimization
numerical integration
linear algebra
Fourier transforms
signal processing
image processing
genetic algorithms
ODE solvers
special functions

numpy:
a powerful N-dimensional array object

- basic linear algebra functions
- basic Fourier transforms
- sophisticated random number capabilities

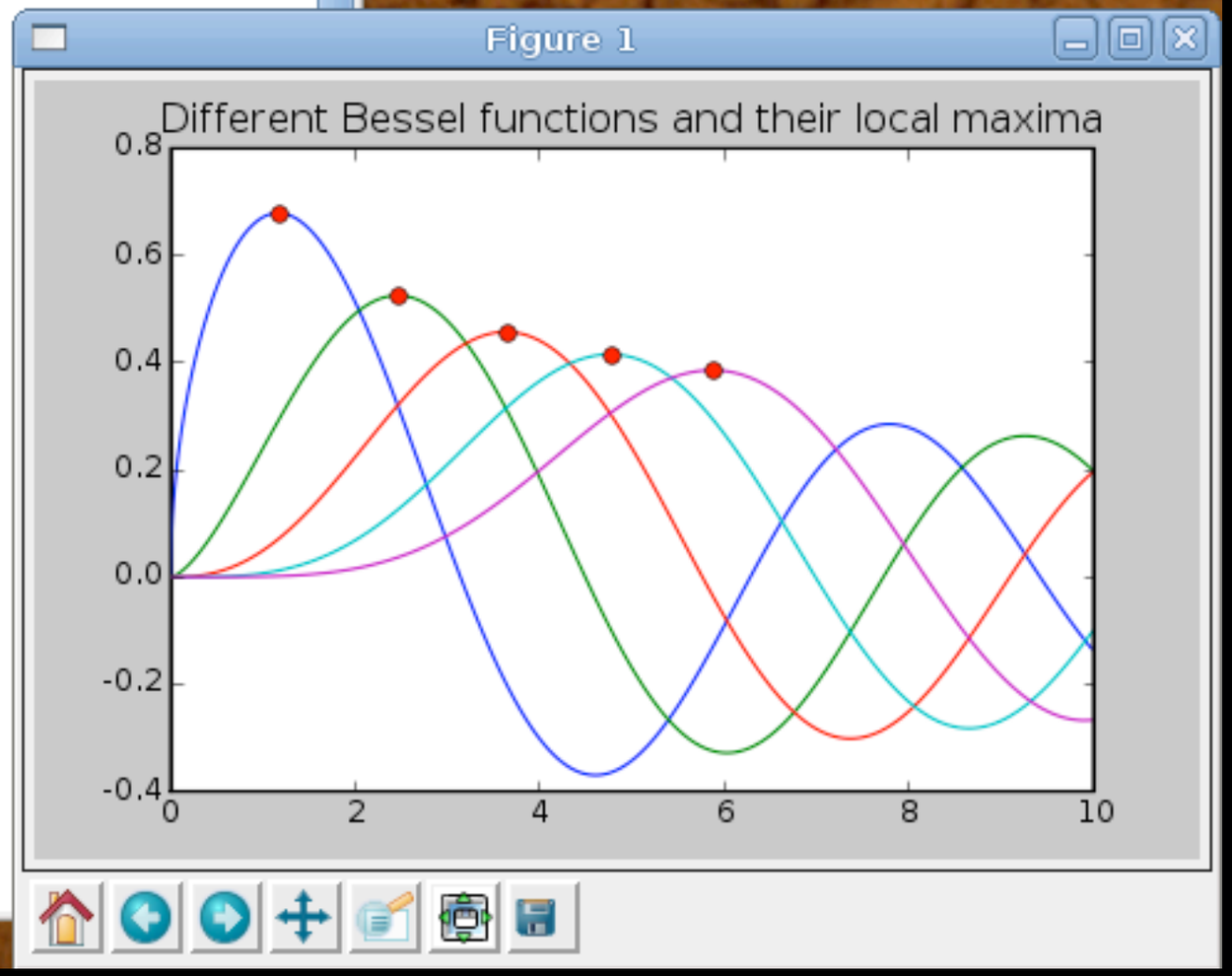
Neuroimaging Pypelines

scipy+numpy

forrest@dolphin: /forrest/work/Math/pywt

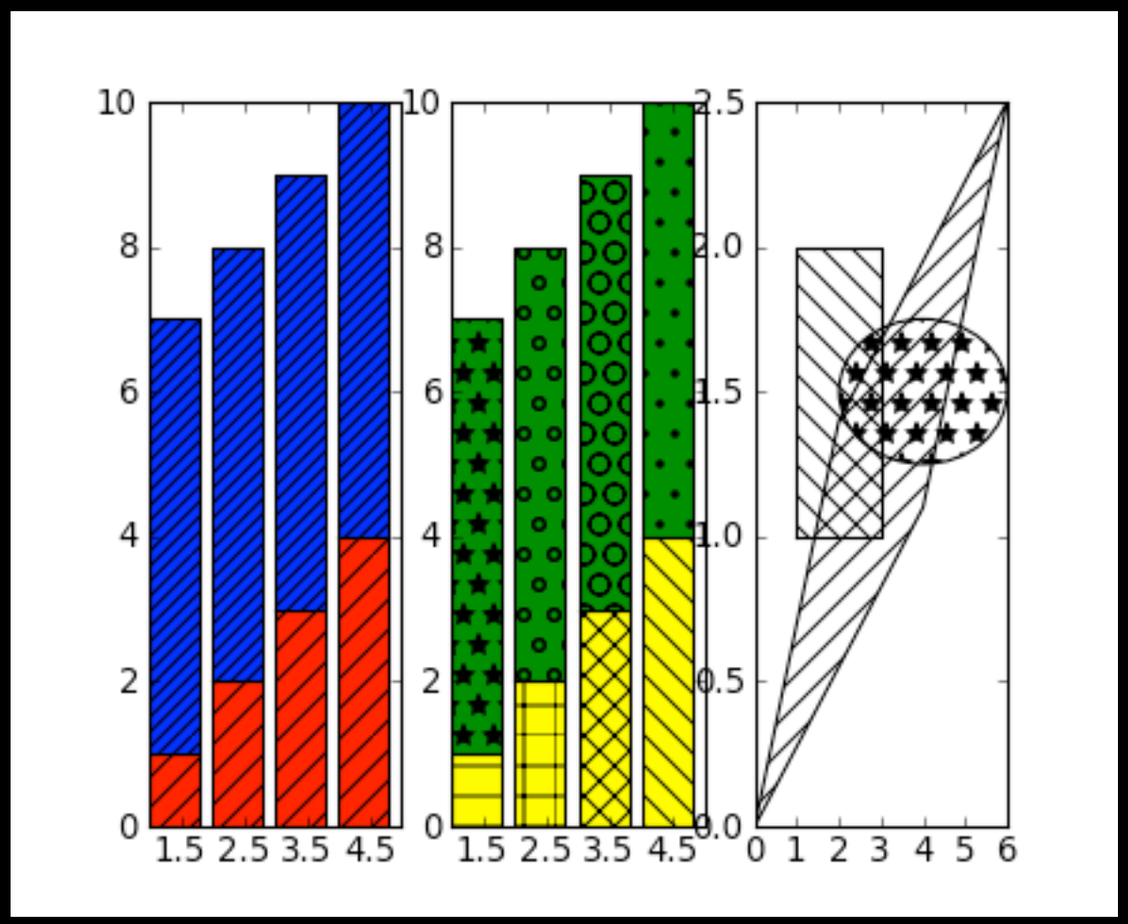
File Edit View Terminal Tabs Help

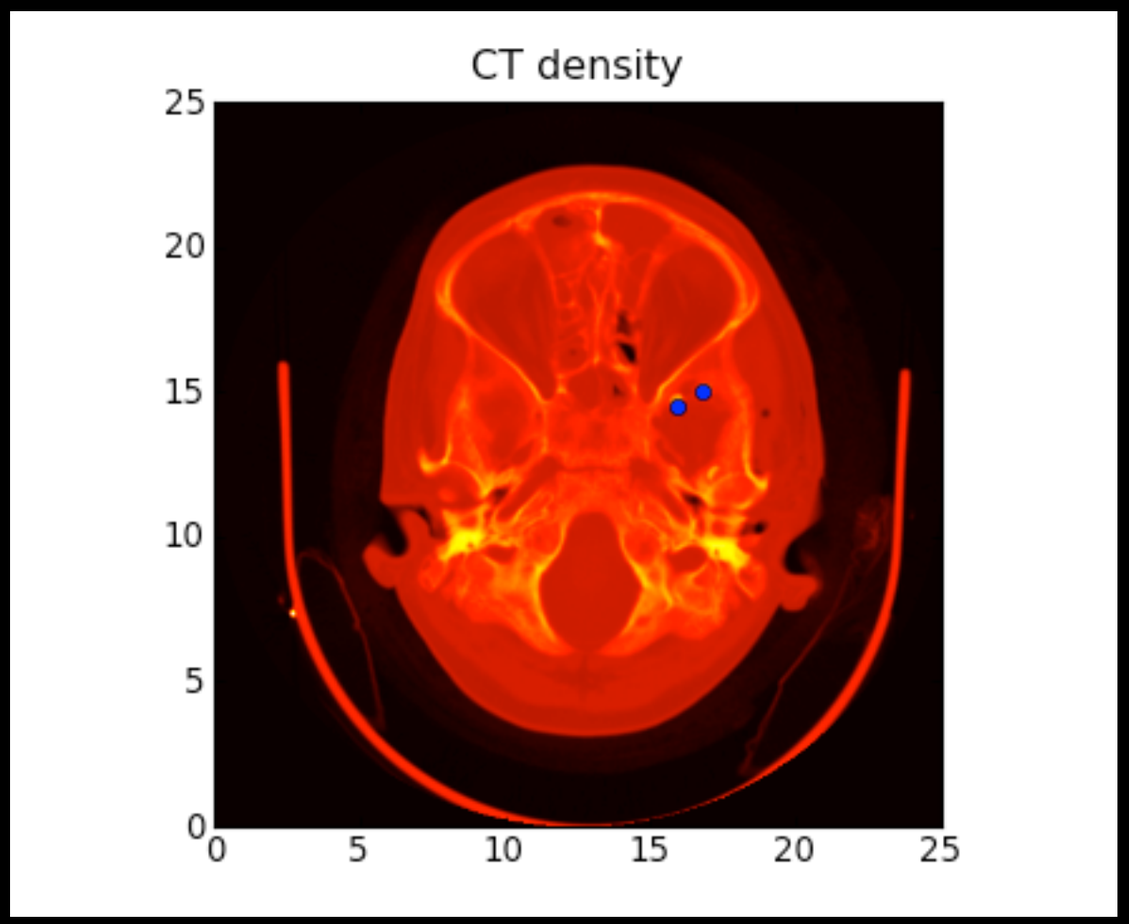
```
Python 2.5.1 (r251:54863, Oct  5 2007, 13:50:07)
[GCC 4.1.3 20070929 (prerelease) (Ubuntu 4.1.2-16ubuntu2)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> from scipy import *
>>> from numpy import *
>>> from pylab import *
>>>
>>> x = arange(0,10,0.01)
>>>
>>> for k in arange(0.5,5.5):
...     y = special.jv(k,x)
...     plot(x,y)
...     f = lambda x: -special.jv(k,x)
...     x_max = optimize.fminbound(f,0,6)
...     plot([x_max], [special.jv(k,x_max)],'ro')
...
[<matplotlib.lines.Line2D instance at 0x1abe368>]
[<matplotlib.lines.Line2D instance at 0x1abe3b0>]
[<matplotlib.lines.Line2D instance at 0x1abe440>]
[<matplotlib.lines.Line2D instance at 0x1abe488>]
[<matplotlib.lines.Line2D instance at 0x1abe4d0>]
[<matplotlib.lines.Line2D instance at 0x1abe518>]
[<matplotlib.lines.Line2D instance at 0x1abe560>]
[<matplotlib.lines.Line2D instance at 0x1abe5a8>]
[<matplotlib.lines.Line2D instance at 0x1abe5f0>]
[<matplotlib.lines.Line2D instance at 0x1abe638>]
>>> title('Different Bessel functions and their local maxima')
<matplotlib.text.Text instance at 0x1abce60>
>>> show()
```



Neuroimaging Pypelines

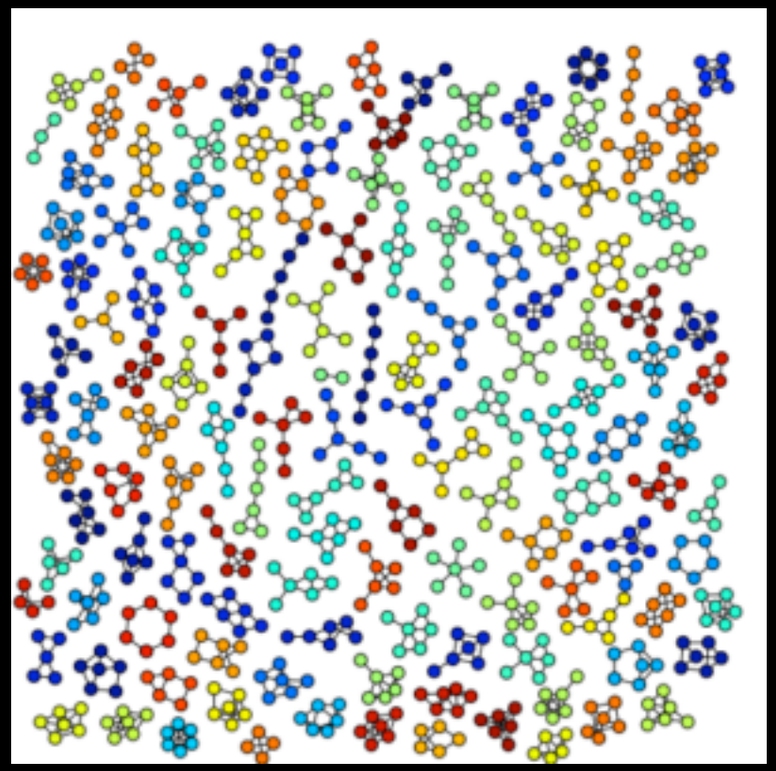
matplotlib

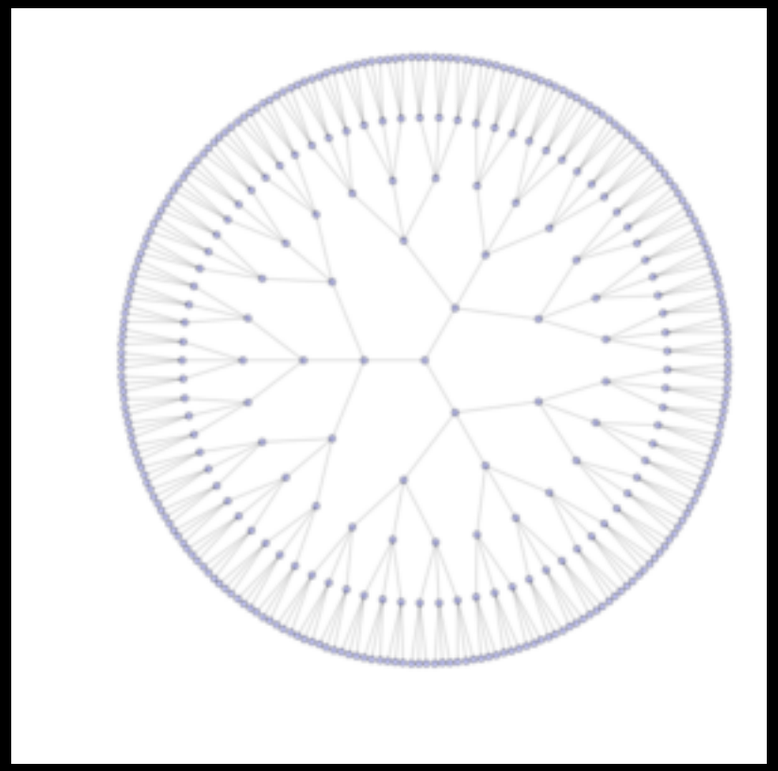


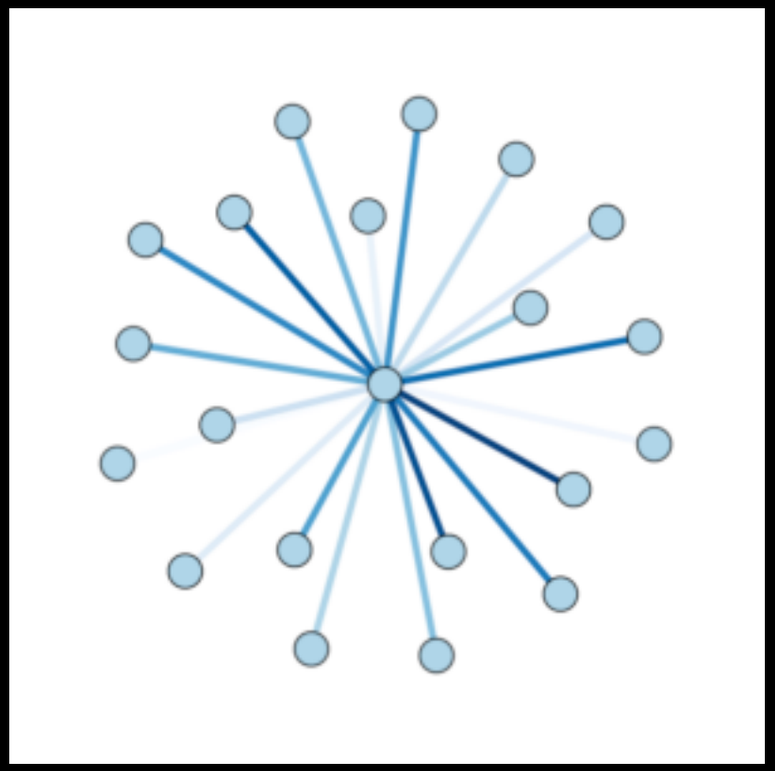


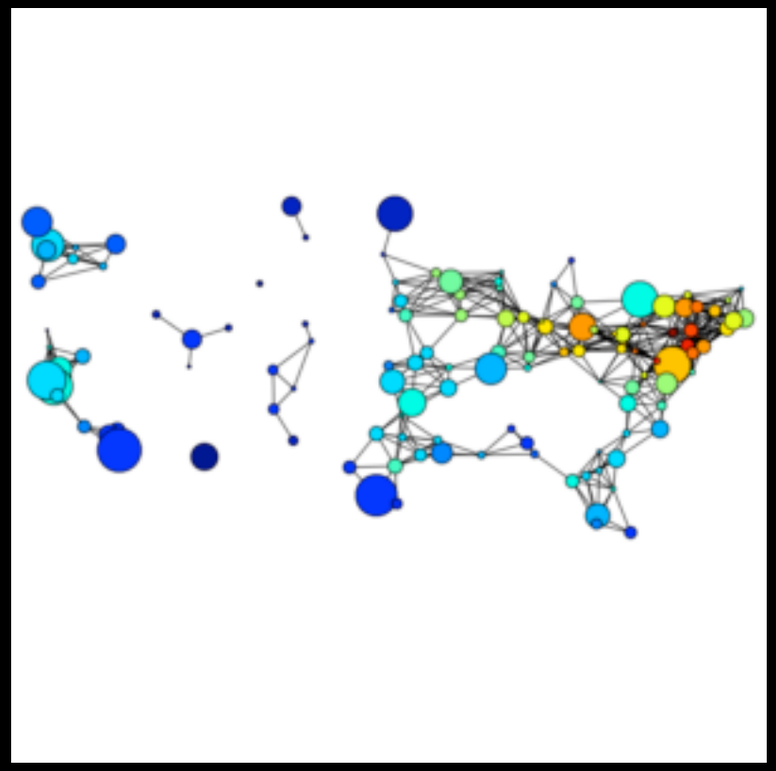
Neuroimaging Pypelines

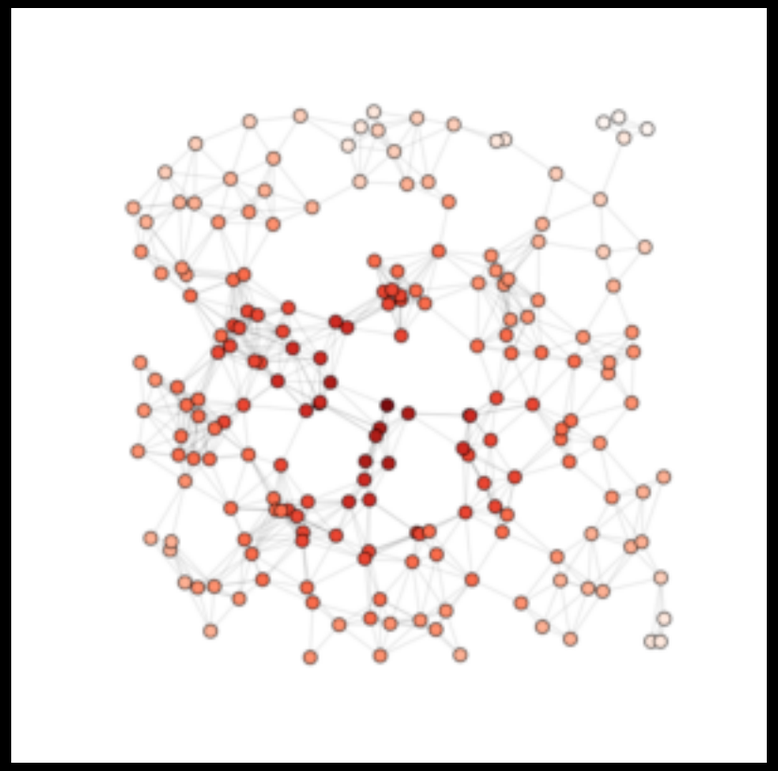
networkx

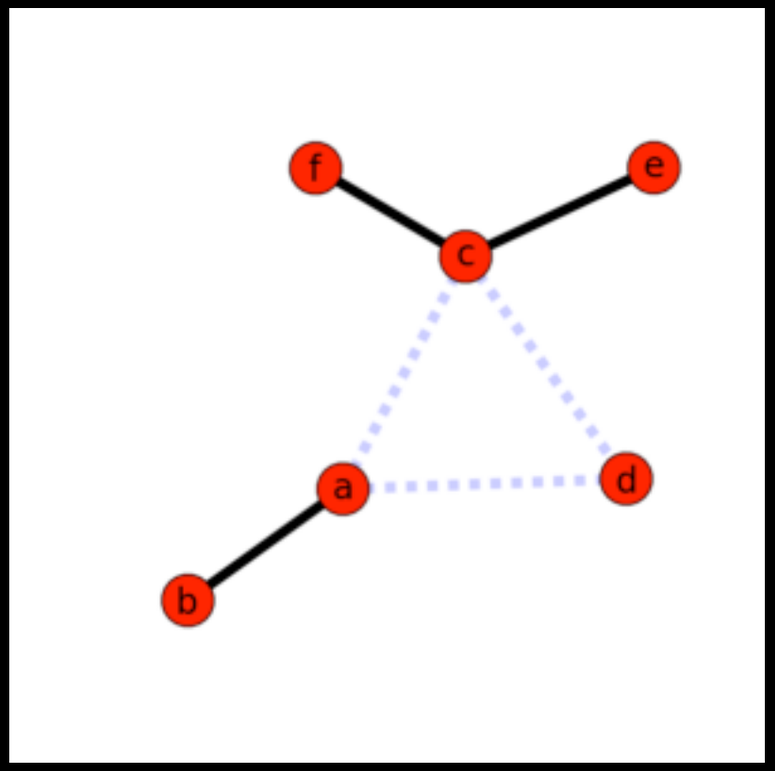






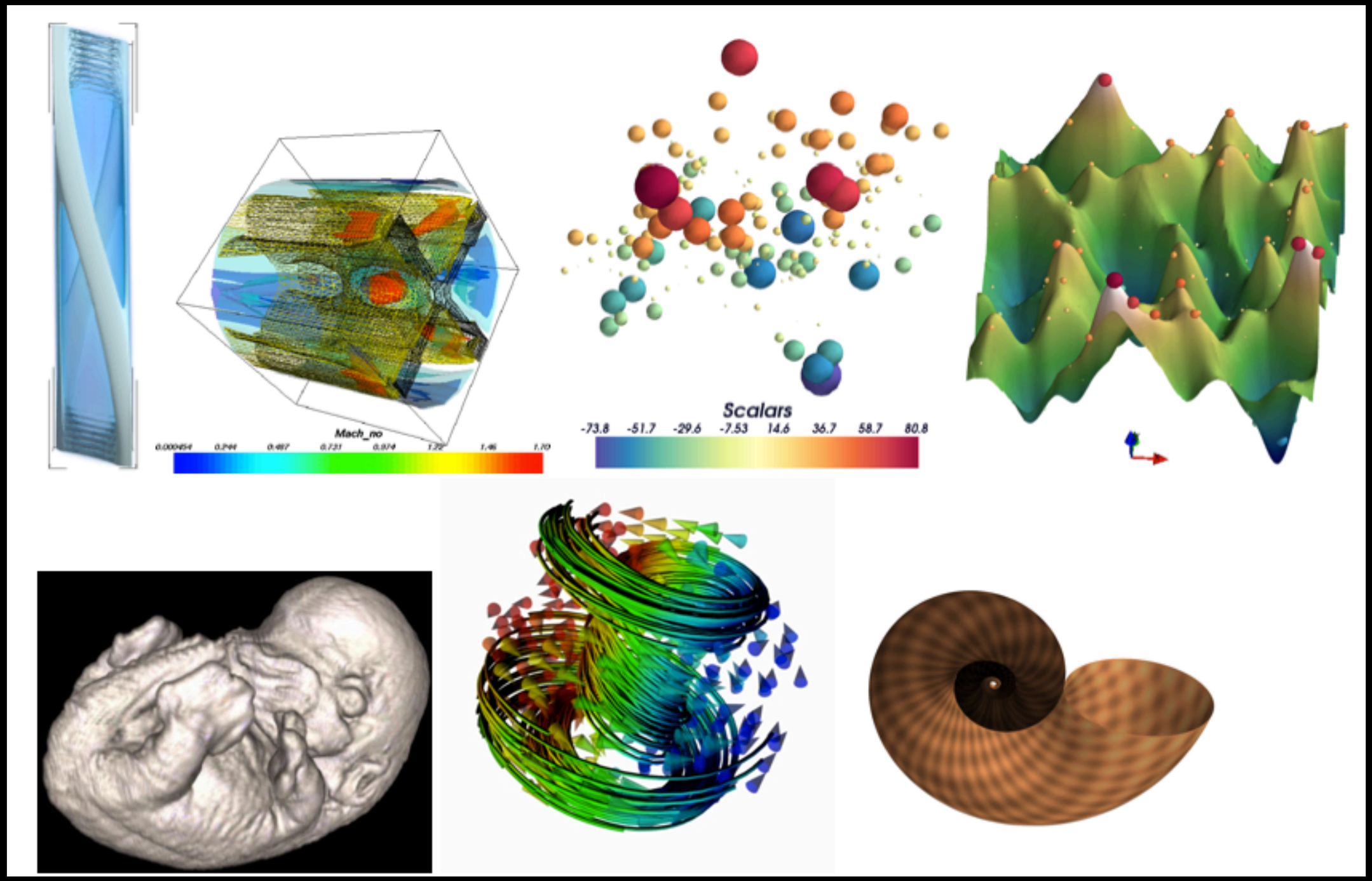






Neuroimaging Pypelines

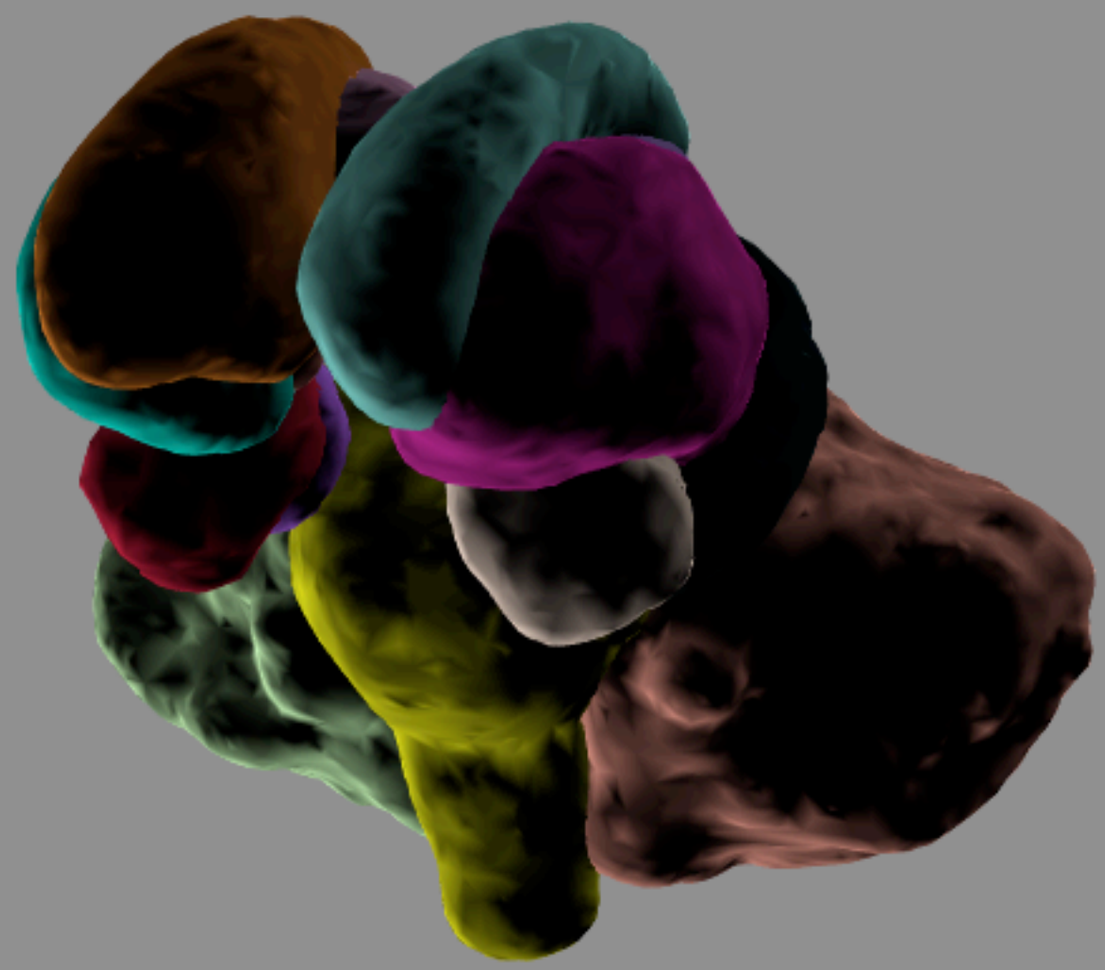
mayavi + tvtk



Neuroimaging Pypelines | mayavi + tvtk

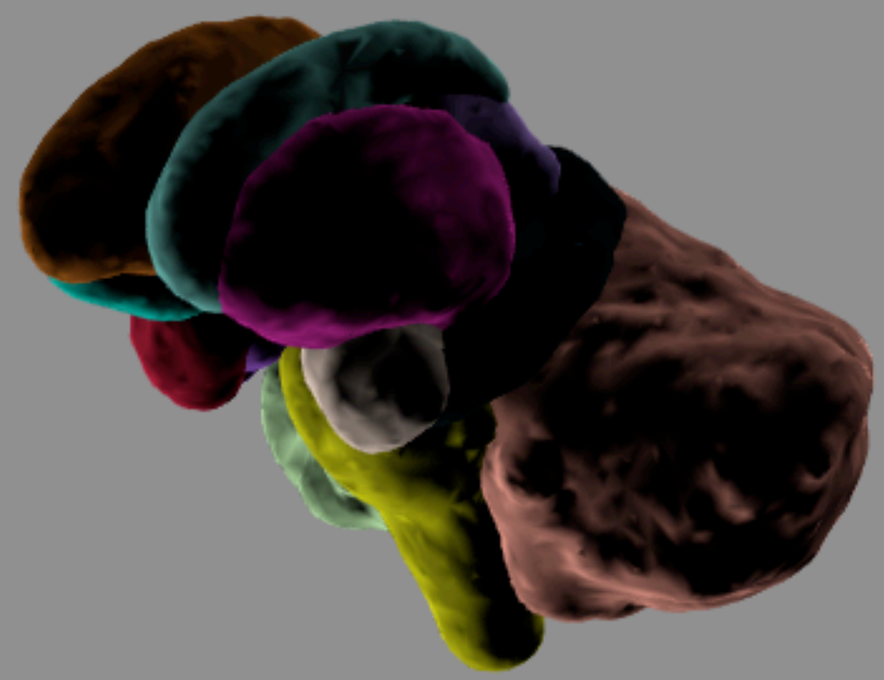
Neuroimaging Pypelines

mayavi + tvtk



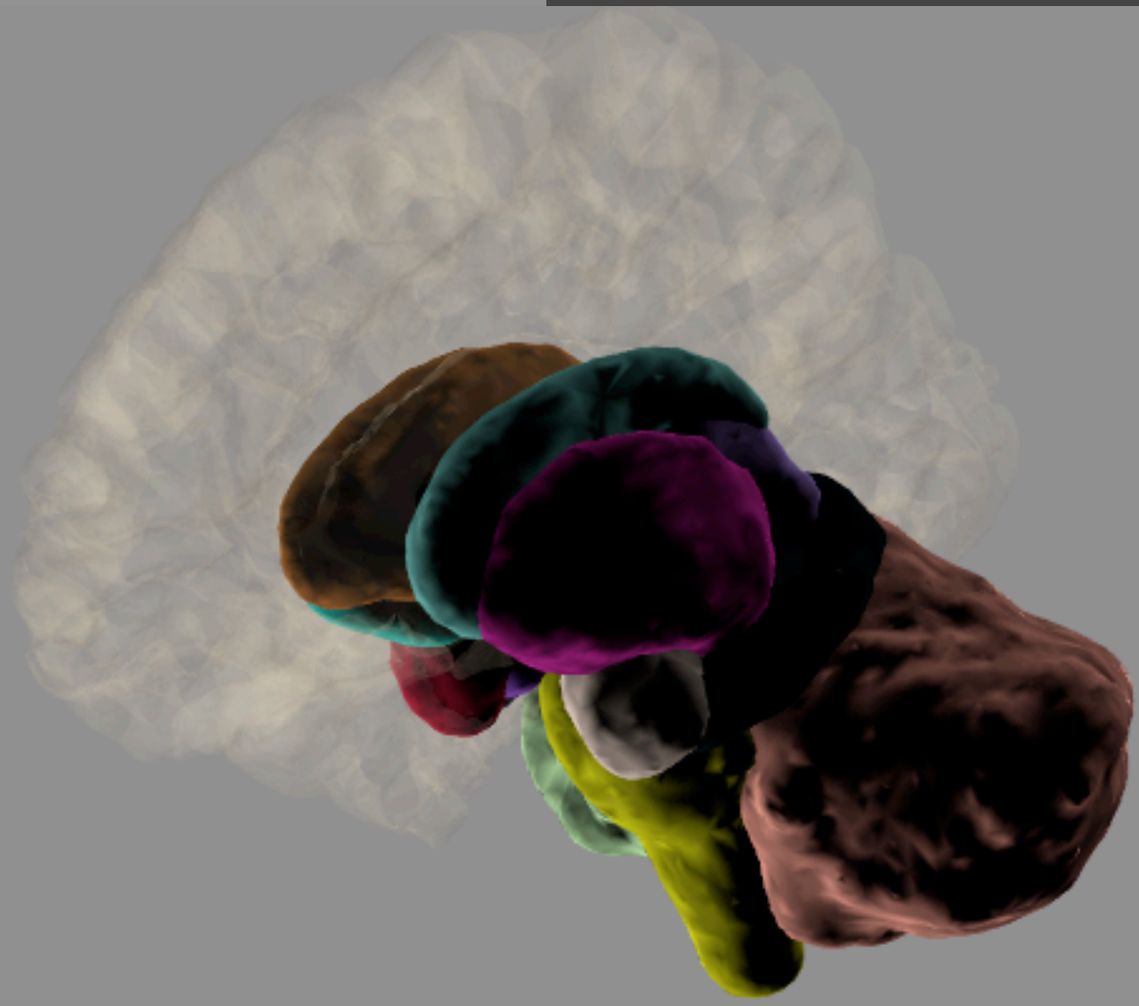
Neuroimaging Pypelines

mayavi + tvtk



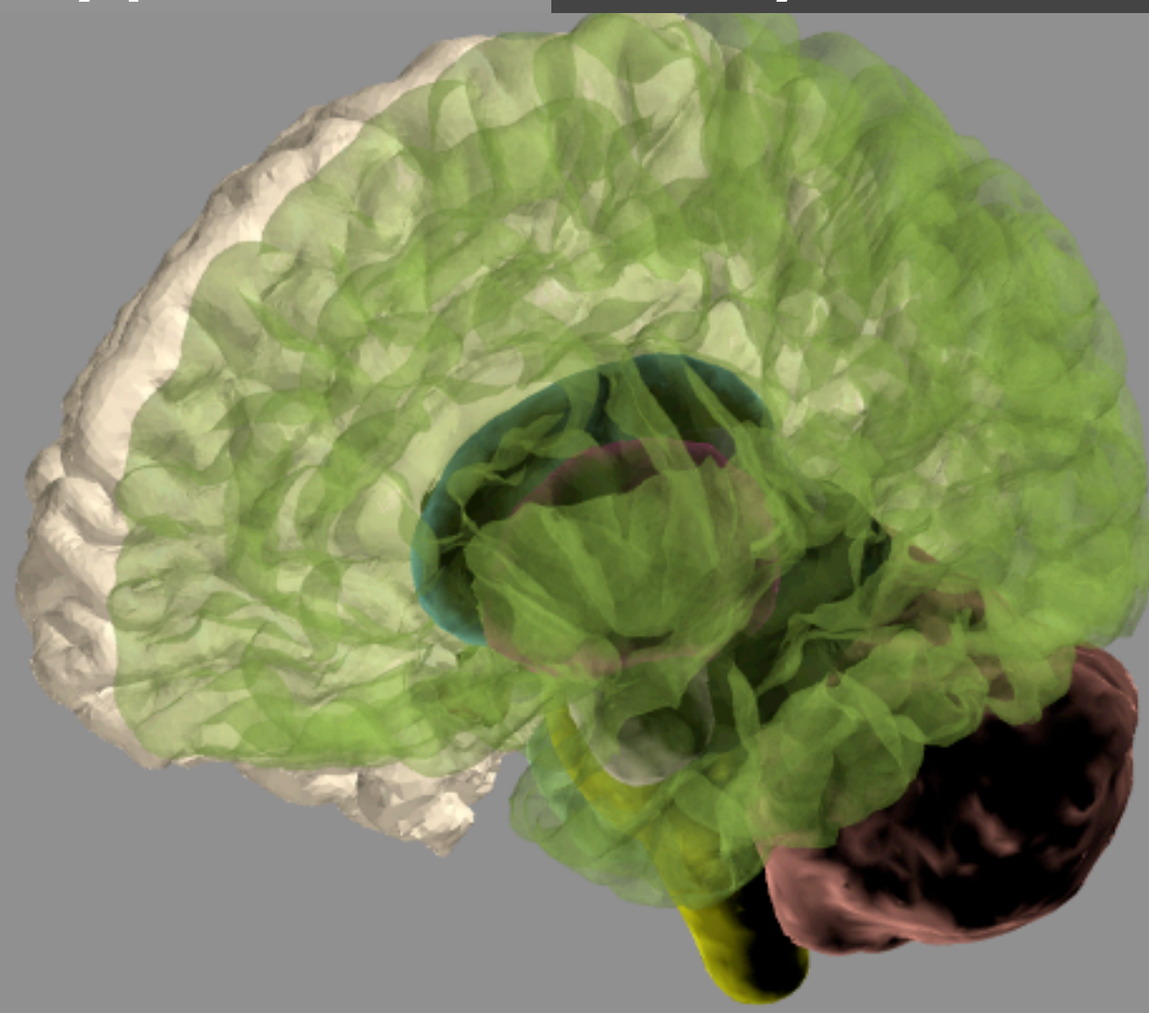
Neuroimaging Pypelines

mayavi + tvtk



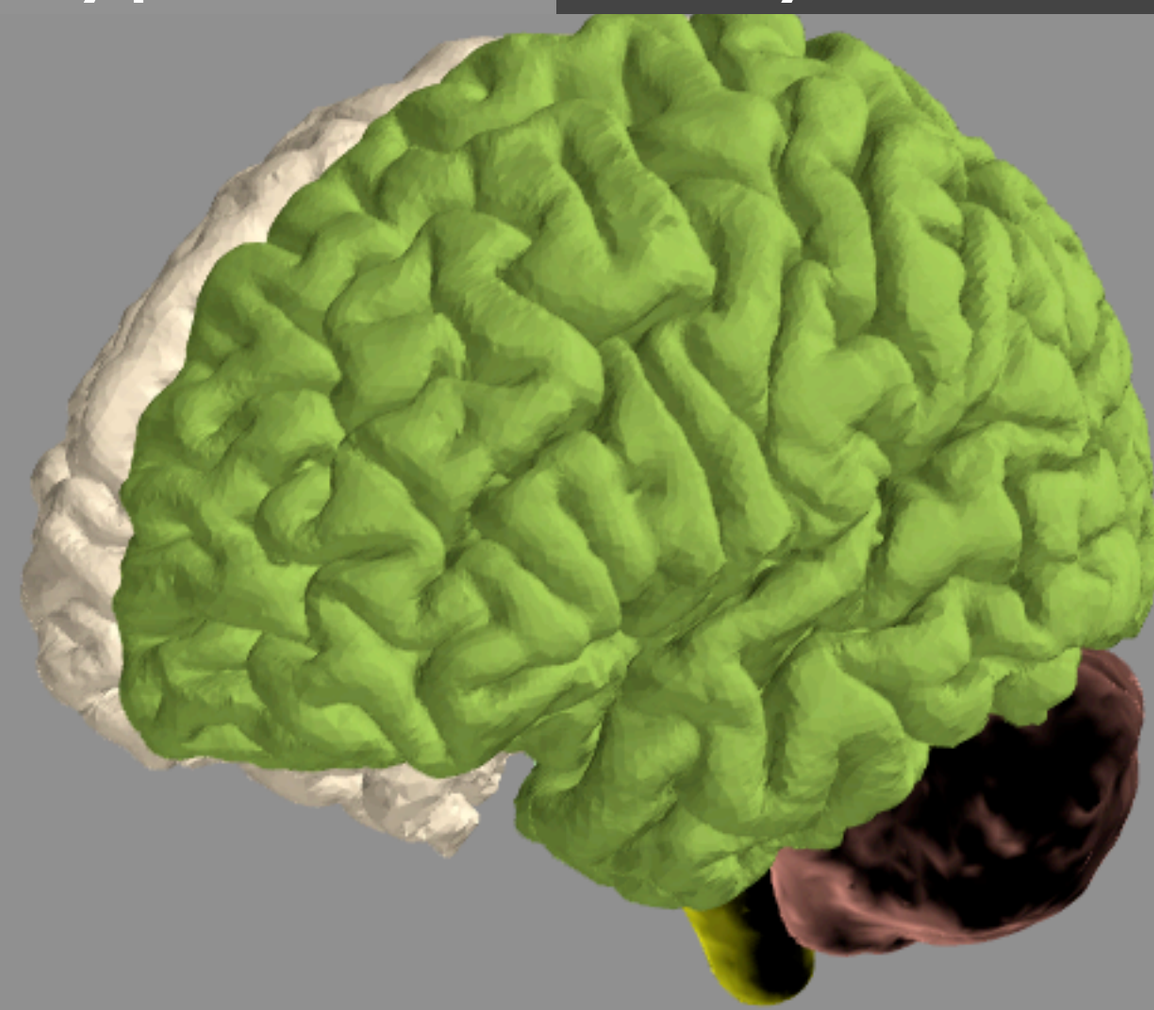
Neuroimaging Pypelines

mayavi + tvtk



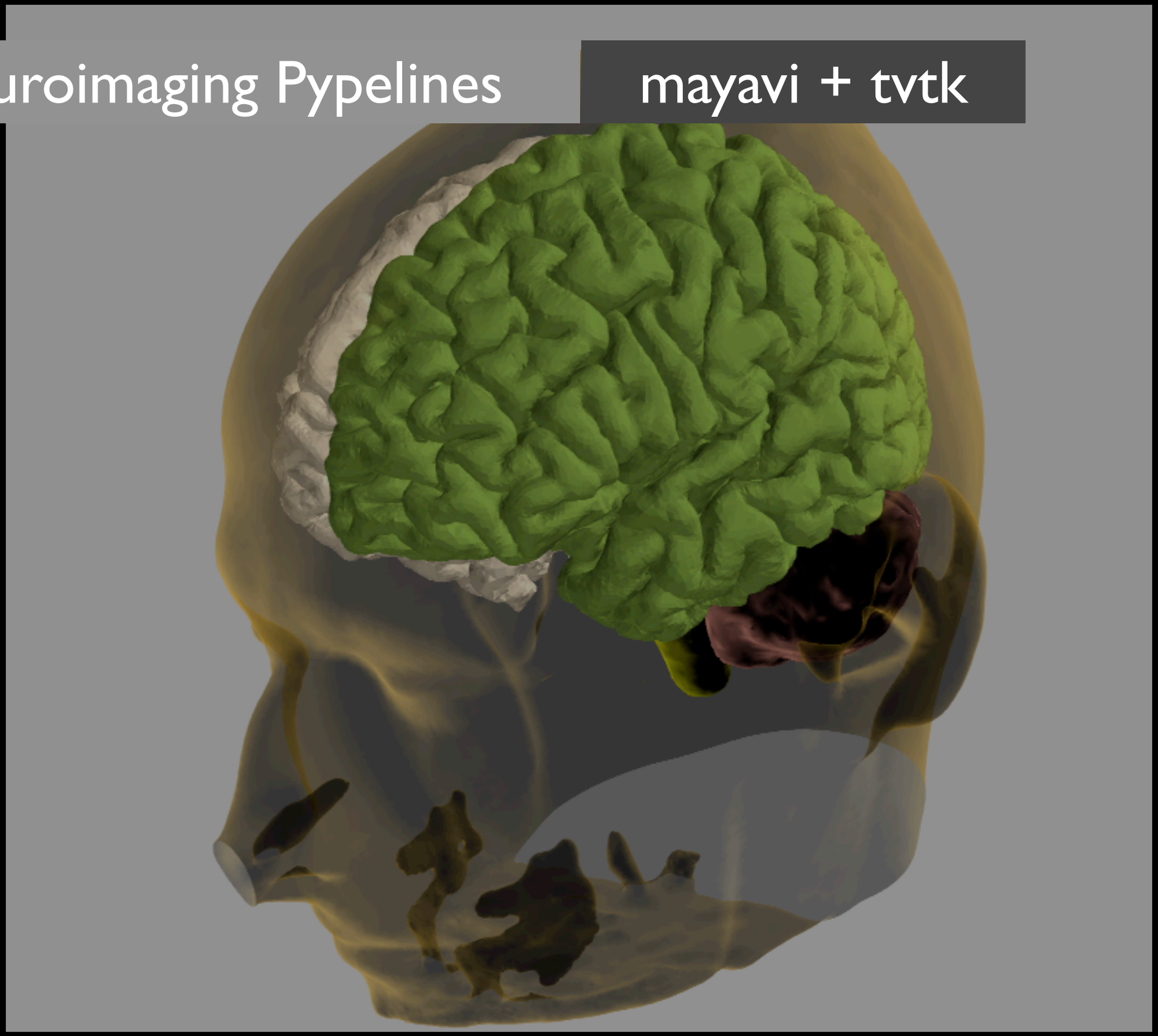
Neuroimaging Pypelines

mayavi + tvtk



Neuroimaging Pypelines

mayavi + tvtk



Neuroimaging Pypelines

mayavi + tvtk

Neuroimaging Pypelines

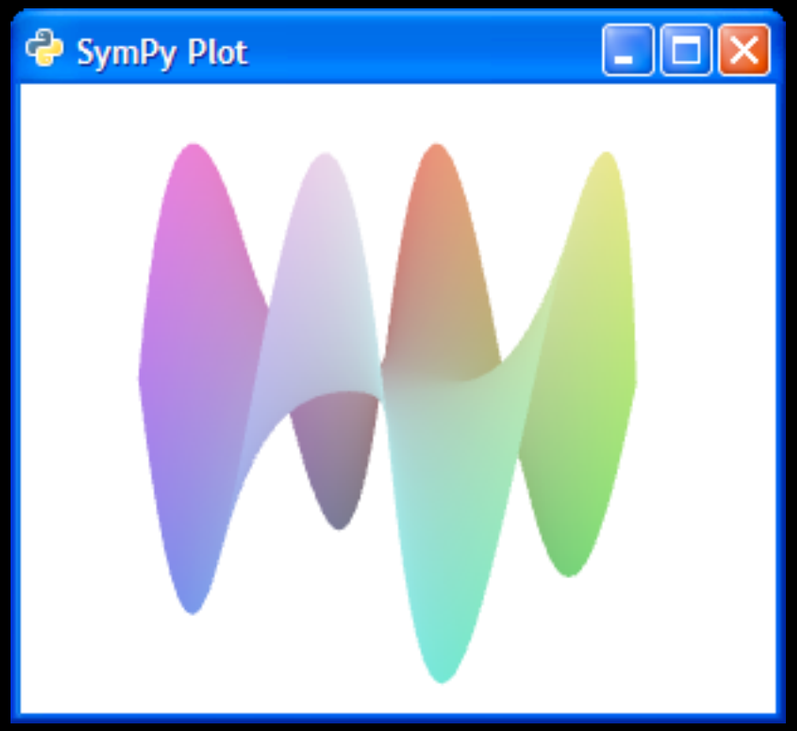
sympy

```
>>> from sympy import *
>>> x = Symbol('x')
>>> diff(sin(x), x)
cos(x)
>>> diff(sin(2*x), x)
2*cos(2*x)

>>> diff(tan(x), x)
1 + tan(x)**2
```

You can check, that it is correct by:

```
>>> limit((tan(x+y)-tan(x))/y, y, 0)
1 + tan(x)**2
```

```
In [5]: Integral((a+b)**2, a).doit()
Out[5]:
 3
α        2        2
── + α*β  + β*α
3

In [6]: sqrt(_)
Out[6]:
                   ____________________
                  ╱  3
                  α        2        2
                  ── + α*β  + β*α
                  3
```



interactive shell with history
but comes with a lot of magic

kernel for distributed computation
(ssh, mpi, torque and threading)

interactive shell with history
but comes with a lot of magic

kernel for distributed computation
(ssh, mpi, torque and threading)

Neuroimaging Pypelines

compneurobiopy

mdp	visionegg	mgl vision
pymvpa	pyschopy	biopython

Home
Neuroscience
Psychology
Psychiatry
Neurology
Journal Info
About the Journal
Editorial Board
Subscribe
Advertise
Journal Content
Archive
Special Topics
Info for Authors
Why publish?
Fees
Article Types
Special Topics
Guidelines
Checklist
Submit Manuscript

SPECIAL TOPIC

Python in neuroscience

Hosted by
Rolf Kötter, rk@donders.ru.nl

Guest editors
James A. Bednar (University of Edinburgh, UK)
Andrew Davison (UNIC, CNRS, France)
Markus Diesmann (RIKEN Brain Science Institute, Japan)
Marc-Oliver Gewaltig (Honda Research Institute Europe GmbH, Germany)
Michael Hines (Yale University, USA)
Eilif Muller (LCN-EPFL, Switzerland)

ABOUT THIS SPECIAL TOPIC

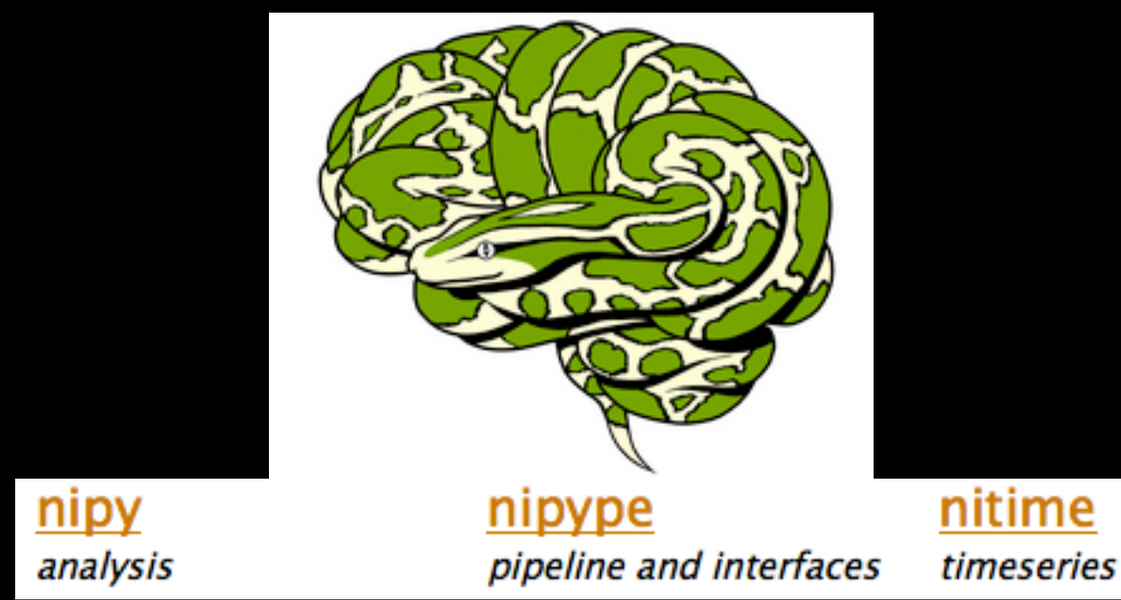
Python is rapidly becoming the de facto standard language for systems integration. Python has a large user and developer-base external to the neuroscience community, and a vast module library that facilitates rapid and maintainable development of complex and intricate systems.

In this special topic, we highlight recent efforts to develop Python modules for the domain of neuroscience software and neuroinformatics:

- simulators and simulator interfaces
- data collection and analysis
- sharing, re-use, storage and databasing of models and data
- stimulus generation
- parameter search and optimization
- visualization
- VLSI hardware interfacing

Moreover, we seek to provide a representative overview of existing mature Python modules for neuroscience and neuroinformatics, to demonstrate a critical mass and show that Python is an appropriate choice of interpreter interface for future neuroscience software development.

Neuroimaging Pypelines | NiPy



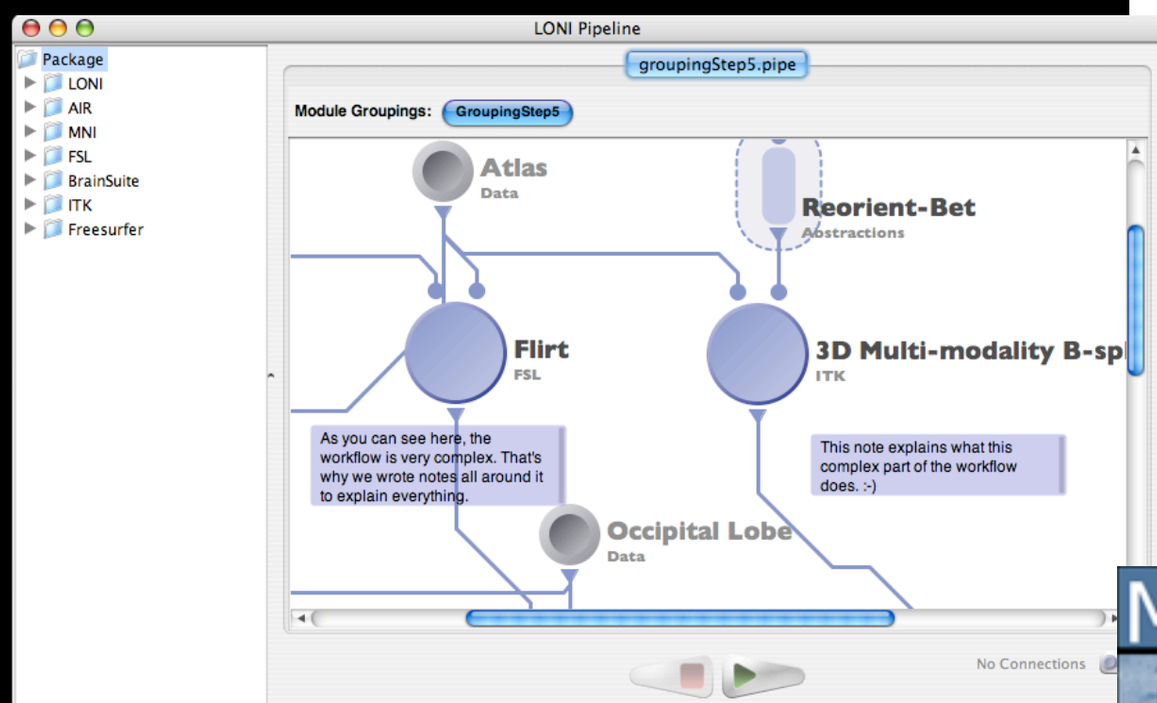
http://nipy.sourceforge.net/

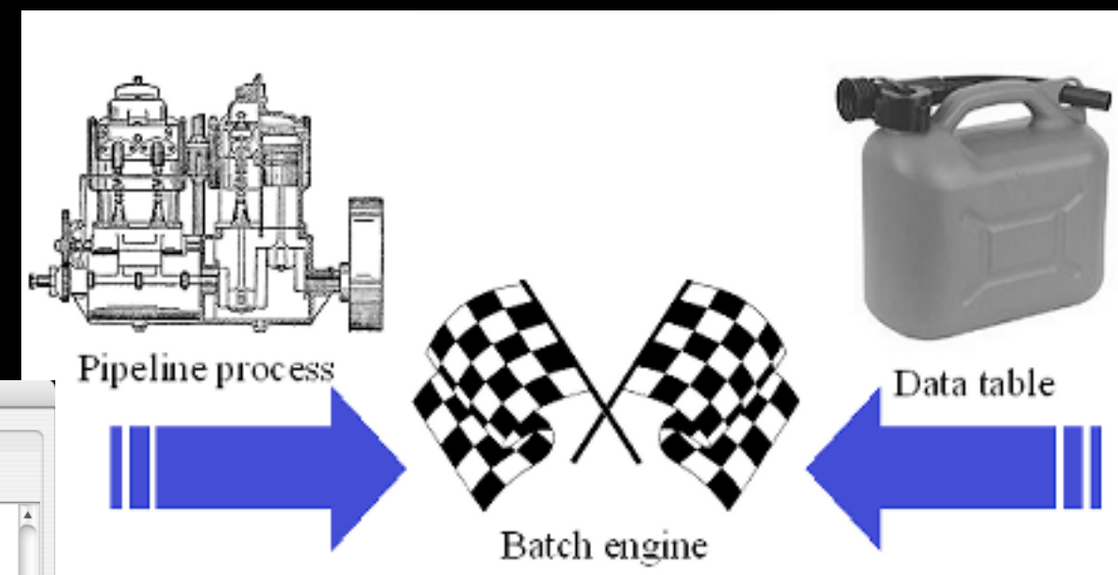
image courtesy dr. arno klein

Neuroimaging Pypelines

why

LONI





Camba

MIPAV

MEDICAL IMAGE PROCESSING AND VISUALIZATION

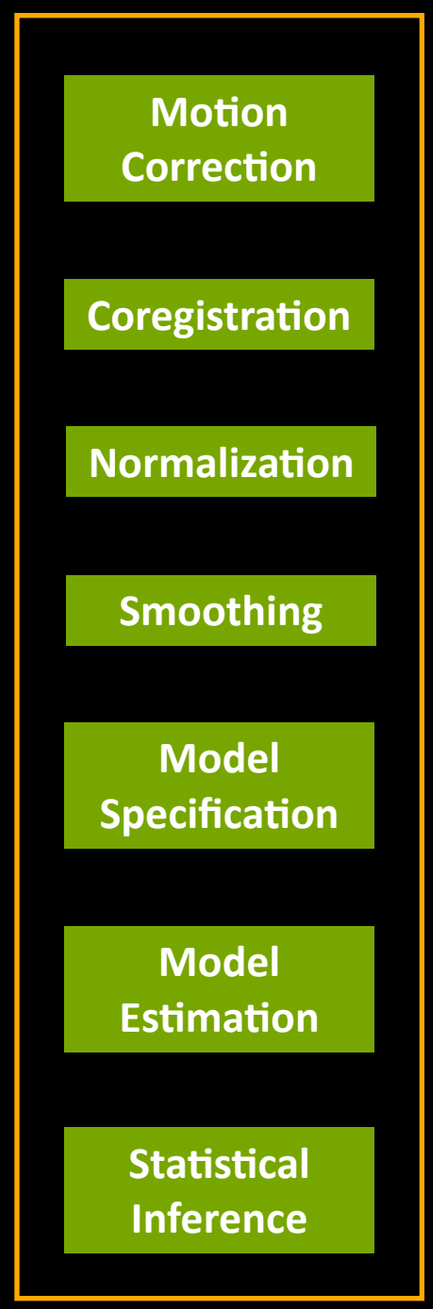
Neuroimaging Pypelines

wishlist

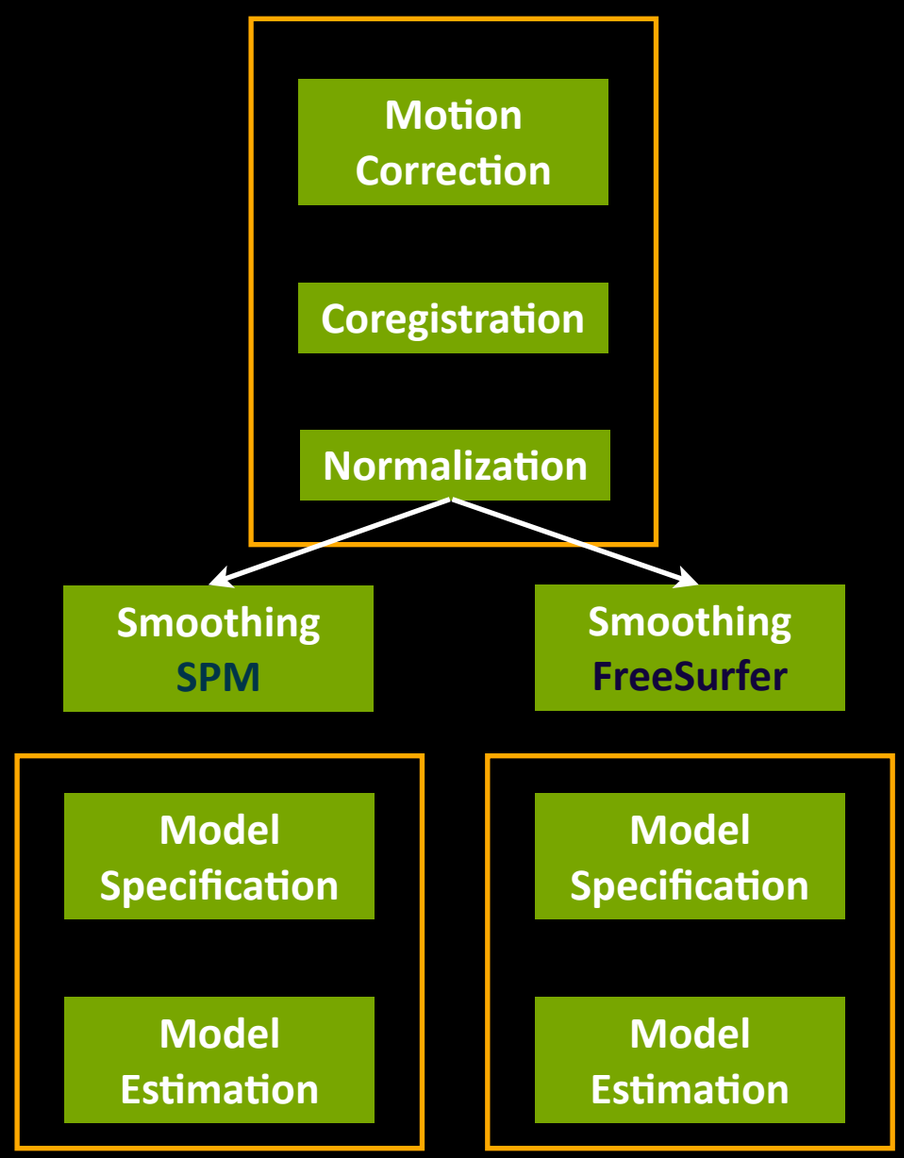
- Motion Correction
- Coregistration
- Normalization
- Smoothing
- Model Specification
- Model Estimation
- Statistical Inference

Neuroimaging Pypelines

wishlist



Compare algorithms



Neuroimaging Pypelines

Motion Correction

Coregistration

Normalization

Smoothing

Model Specification

Model Estimation

Statistical Inference

wishlist

Compare algorithms

Motion Correction

Coregistration

Normalization

Smoothing SPM

Model Specification

Model Estimation

Smoothing FreeSurfer

Model Specification

Model Estimation

Custom Software

Preprocessing

ART: Quality Assurance

Model Specification

Model Estimation

Statistical Inference

Neuroimaging Pypelines

wishlist

Motion Correction

Coregistration

Normalization

Smoothing

Model Specification

Model Estimation

Statistical Inference

Compare algorithms

Motion Correction

Coregistration

Normalization

Smoothing SPM

Smoothing FreeSurfer

Model Specification

Model Estimation

Model Specification

Model Estimation

Custom Software

Preprocessing

ART: Quality Assurance

Model Specification

Model Estimation

Statistical Inference

Optimized processing

Motion Correction FSL

Coregistration SPM

Normalization ANTS

Smoothing FreeSurfer

Model Specification NIPYPE

Model Estimation NIPY

goals (part ii)

- Lightweight
- Open-source, community supported and developed
- Extensible (plugin) framework based on a high-level language
- Integrates different neuroimaging software (FSL, SPM, etc.,.)
- Interoperability (including SPM)
- Distributed, non-redundant processing
- Minimize data redundancy
- Bring consistency, repeatability and clarity into analysis
- Teaching tool

Target three levels of users

drag and droppers
scripters
coders

Target three levels of users

~~drag and droppers~~

scripters

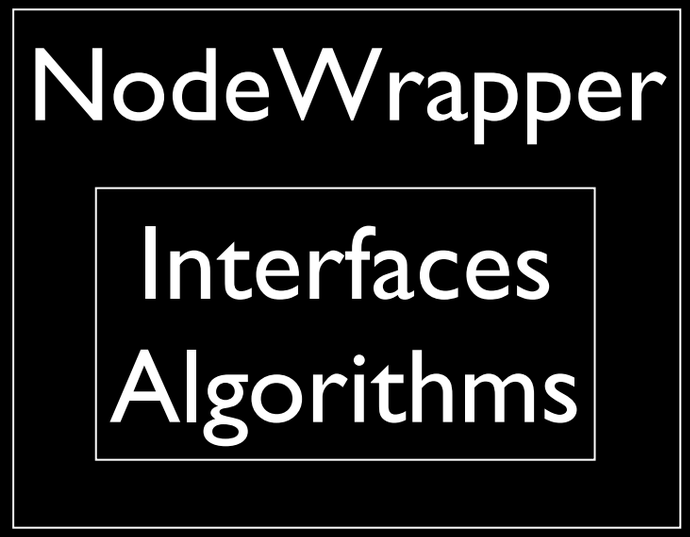
coders

Neuroimaging Pypelines | architecture

Neuroimaging Pypelines | architecture

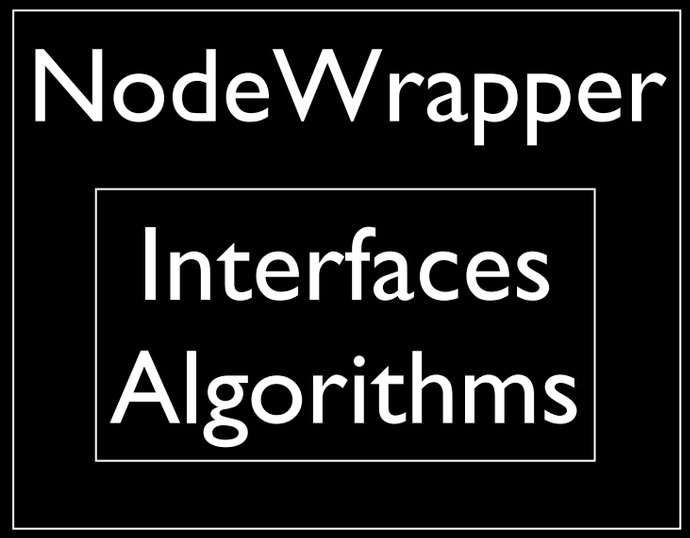
Interfaces
Algorithms

Interfaces
Algorithms



Neuroimaging Pypelines | architecture

Interfaces
Algorithms



Interfaces + Algorithms

- Provides wrappers/interfaces to call external software (e.g., SPM, FSL)
- Each process defines its own inputs and outputs
- Provides information about the execution of the underlying process

Interfaces + Algorithms

- Provides wrappers/interfaces to call external software (e.g., SPM, FSL)
- Each process defines its own inputs and outputs
- Provides information about the execution of the underlying process

```
Interface
|-- CommandLine
    |-- MatlabCommandLine
```

Example. Interfaces

```
import nipype.interfaces.fsl as fsl
mybet = fsl.Bet(infile='foo.nii', outfile='bar.nii')
result = mybet.run()
```

```
import nipype.interfaces.fsl as fsl
mybet = fsl.Bet()
mybet.inputs.infile = 'foo.nii'
mybet.inputs.outfile = 'bar.nii'
result = mybet.run()
```

```
import nipype.interfaces.fsl as fsl
mybet = fsl.Bet()
result = mybet.run(infile='foo.nii', outfile='bar.nii', frac=0.5)
```

```
import nipype.interfaces.fsl as fsl
realigner = fsl.McFlirt()
realigner.infile='timeseries4D.nii'
result = realigner.run()
```

```
import nipype.interfaces.spm as spm
from glob import glob
allepi = glob('epi*.nii') # this will return an unsorted list
allepi.sort()
realigner = spm.Realign()
realigner.inputs.infile = allepi
result = realigner.run()
```

Neuroimaging Pypelines inventory

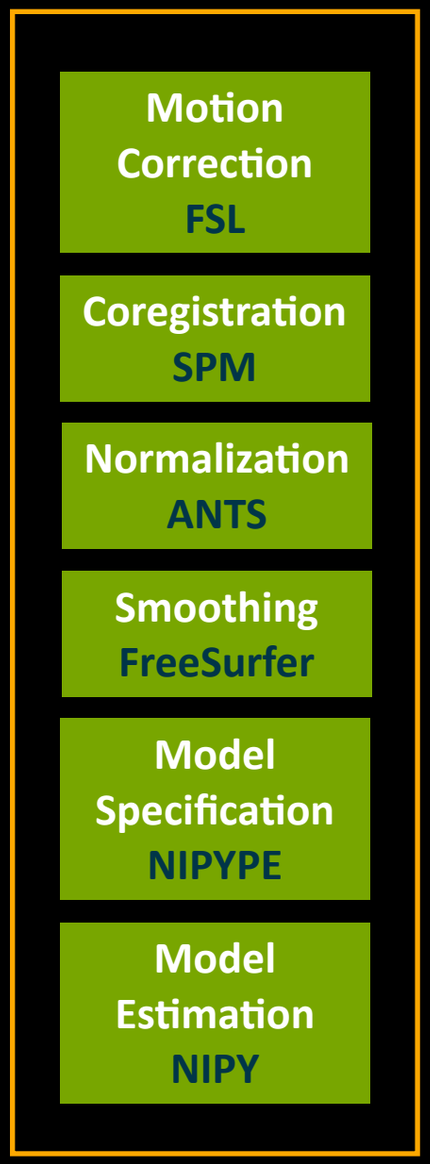
- interfaces.spm
  - Module: `interfaces.spm`
  - Classes
    - `Coregister`
    - `EstimateContrast`
    - `EstimateModel`
    - `Level1Design`
    - `Normalize`
    - `OneSampleTTest`
    - `Realign`
    - `Segment`
    - `Smooth`
    - `SpecifyModel`
    - `SpmInfo`
    - `SpmMatlabCommandLine`
    - `TwoSampleTTest`

- interfaces.freesurfer
  - Module: `interfaces.freesurfer`
    - Examples
  - Classes
    - `ApplyVolTransform`
    - `BBRegister`
    - `Dicom2Nifti`
    - `FSCommandLine`
    - `OneSampleTTest`
    - `ReconAll`
    - `Resample`
    - `Smooth`
    - `SurfConcat`
    - `Threshold`

- algorithms.modelgen
  - Module: `algorithms.modelgen`
  - `SpecifyModel`
- algorithms.rapidart
  - Module: `algorithms.rapidart`
  - Classes
    - `ArtifactDetect`
    - `StimulusCorrelation`

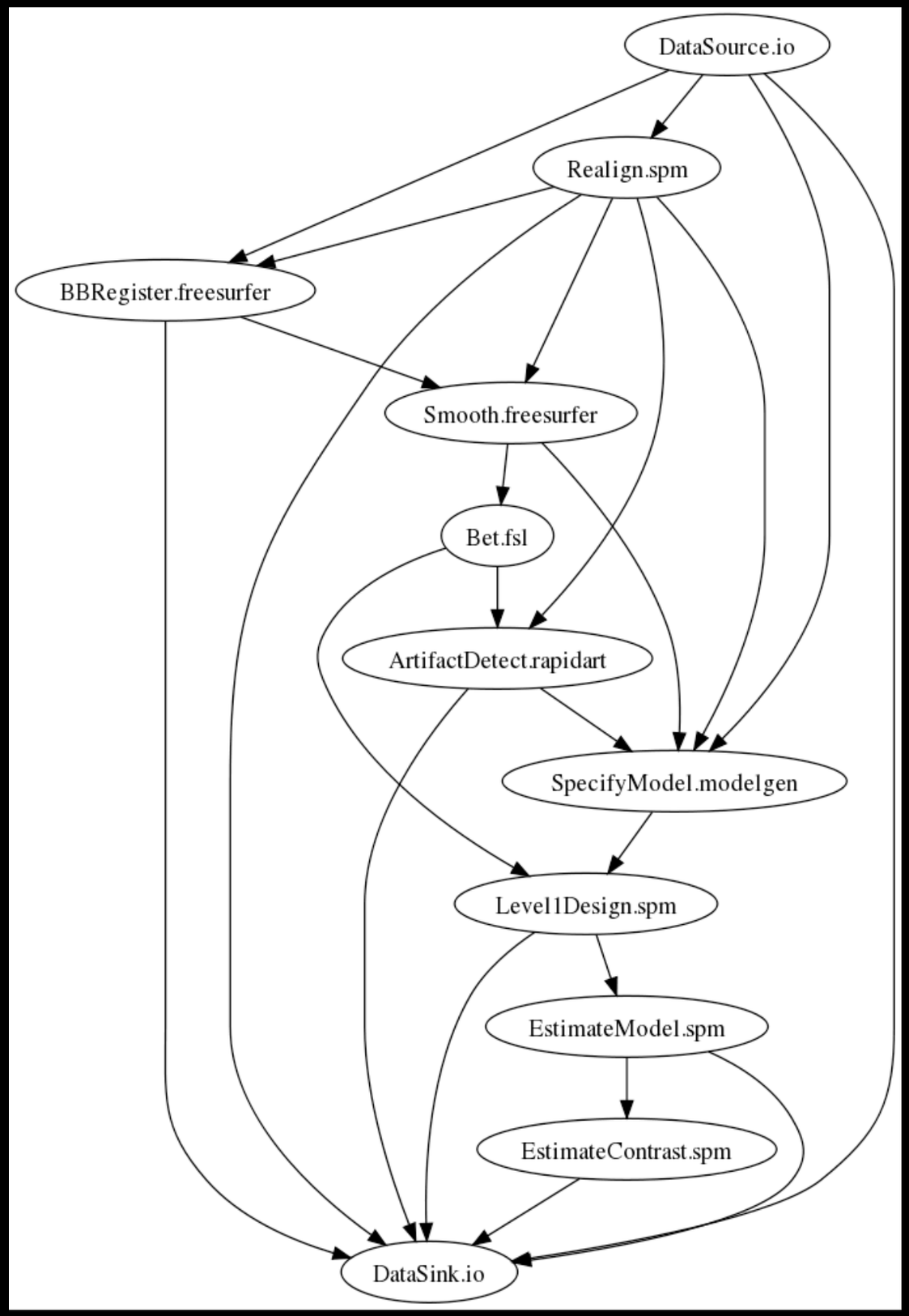
- interfaces.fsl
  - Module: `interfaces.fsl`
    - Examples
  - Classes
    - `ApplyWarp`
    - `ApplyXFM`
    - `Bedpostx`
    - `Bet`
    - `Dtifit`
    - `Eddy_correct`
    - `FSLCommand`
    - `FSLInfo`
    - `FSLSmooth`
    - `Fast`
    - `Find_the_biggest`
    - `Flirt`
    - `Fnirt`
    - `Fslmaths`
    - `Fslroi`
    - `L1FSFmaker`
    - `Level1Design`
    - `McFlirt`
    - `Probtrackx`
    - `Proj_thresh`
    - `Randomise`
    - `Randomise_parallel`
    - `Tbss1preproc`
    - `Tbss2reg`

Optimized processing



Neuroimaging Pypelines

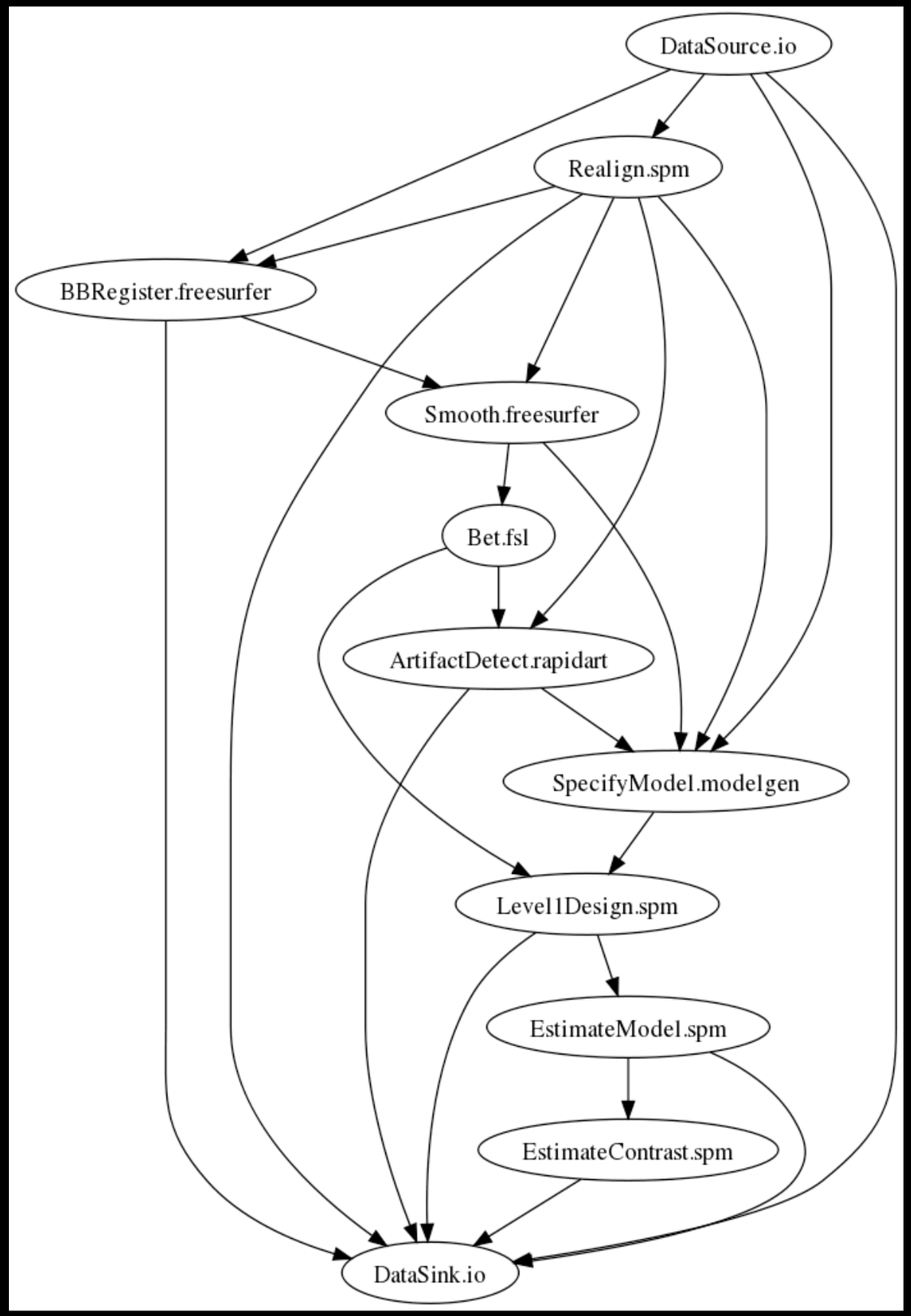
the pipeline



Neuroimaging Pypelines

the pipeline

- Processing pipeline represented as a directed acyclic graph (DAG)
- Leverages graph algorithms for scheduling
- Each process defines its own inputs and outputs and wraps external software (SPM, FSL)
- Separates data specification and code execution
- Can iterate pipeline-segment over parameterizations of modules



Neuroimaging Pypelines

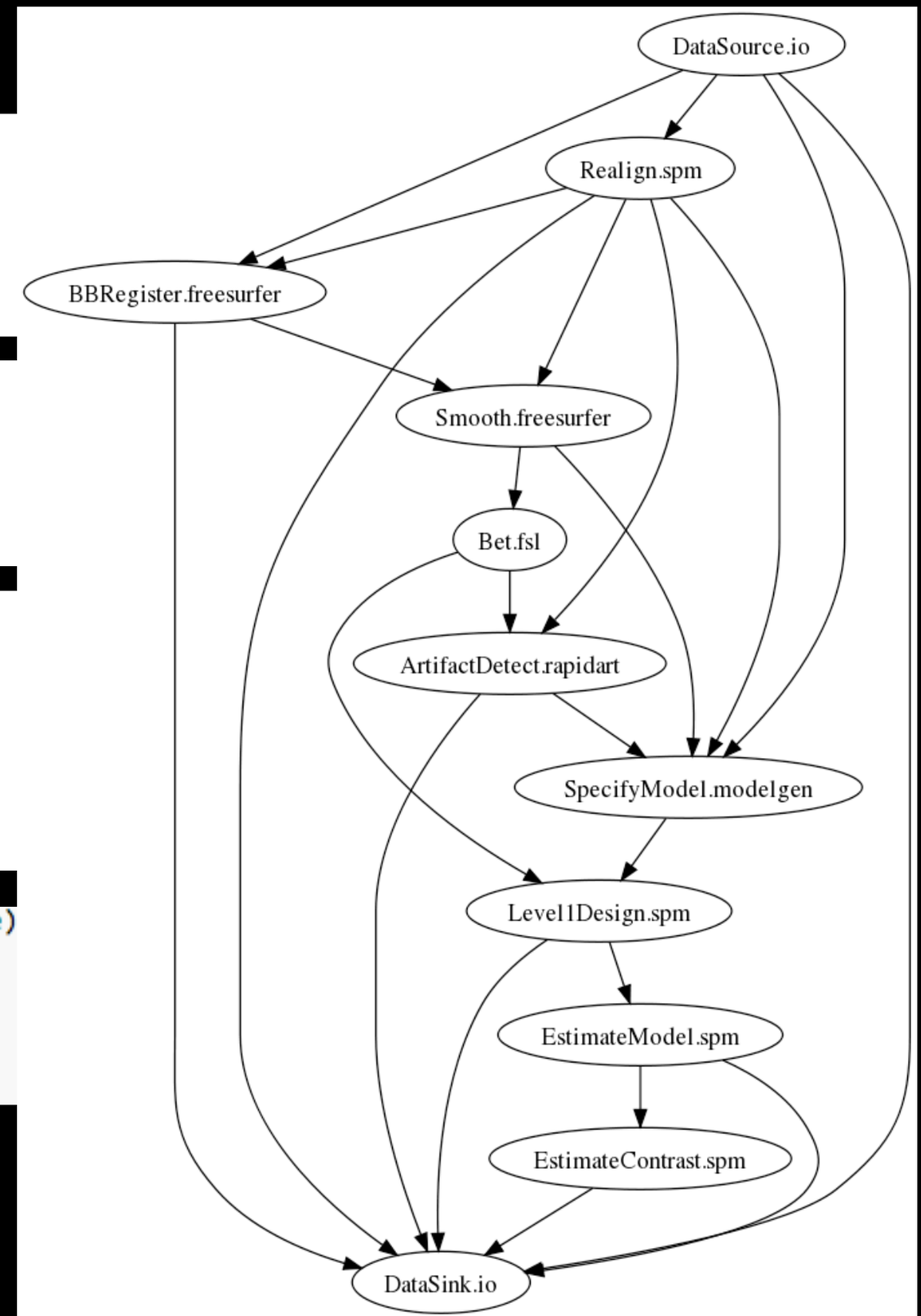
nodes

```python
"""
   c. Use :class:`nipype.interfaces.spm.Realign` for motion correction
   and register all images to the mean image.
"""
realign = nw.NodeWrapper(interface=spm.Realign(),diskbased=True)
realign.inputs.register_to_mean = True
```

```python
"""
   e. Use :class:`nipype.interfaces.fsl.Bet` for skull strip
   structural images.
"""
skullstrip = nw.NodeWrapper(interface=fsl.Bet(),diskbased=True)
skullstrip.inputs.mask = True
```

```python
"""
   h.2. Use :class:`nipype.interfaces.fs.Smooth` to smooth the
   functional data.
"""
smooth = nw.NodeWrapper(interface=fs.Smooth(),diskbased=True)
smooth.inputs.surface_fwhm = 5
smooth.inputs.vol_fwhm     = 6
smooth.iterfield = ['sourcefile']
```

```python
modelspec = nw.NodeWrapper(interface=model.SpecifyModel(),diskbased=True)
modelspec.inputs.concatenate_runs        = True
modelspec.inputs.input_units             = 'secs'
modelspec.inputs.output_units            = 'secs'
modelspec.inputs.time_repetition         = 3.
modelspec.inputs.high_pass_filter_cutoff = 120
```



Neuroimaging Pypelines

the wrapper

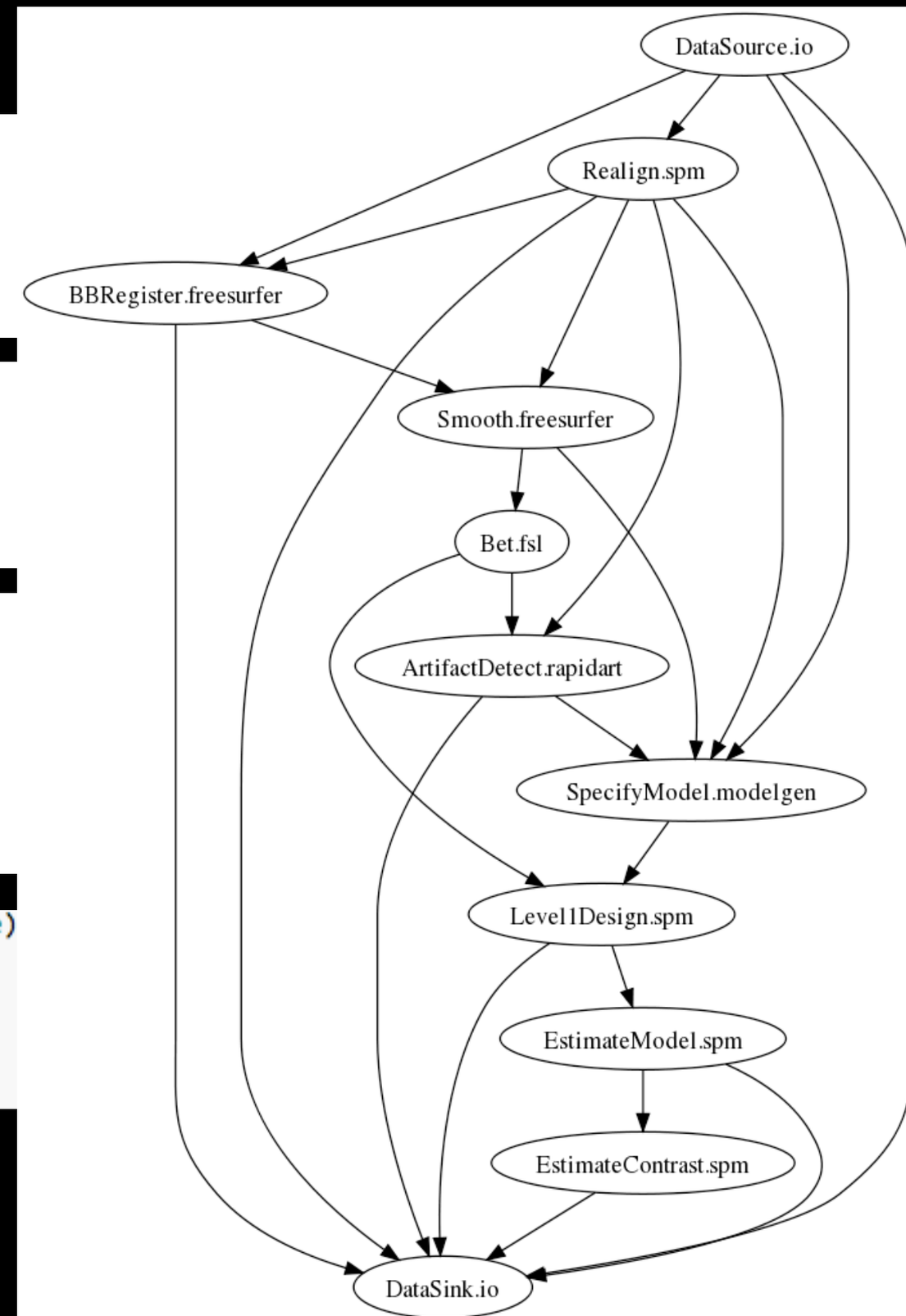
```
"""
   c. Use :class:`nipype.interfaces.spm.Realign` for motion correction
   and register all images to the mean image.
"""
realign = nw.NodeWrapper(interface=spm.Realign(),diskbased=True)
realign.inputs.register_to_mean = True
```

```
"""
   e. Use :class:`nipype.interfaces.fsl.Bet` for skull strip
   structural images.
"""
skullstrip = nw.NodeWrapper(interface=fsl.Bet(),diskbased=True)
skullstrip.inputs.mask = True
```

```
"""
   h.2. Use :class:`nipype.interfaces.fs.Smooth` to smooth the
   functional data.
"""
smooth = nw.NodeWrapper(interface=fs.Smooth(),diskbased=True)
smooth.inputs.surface_fwhm = 5
smooth.inputs.vol_fwhm     = 6
smooth.iterfield = ['sourcefile']
```

```
modelspec = nw.NodeWrapper(interface=model.SpecifyModel(),diskbased=True)
modelspec.inputs.concatenate_runs        = True
modelspec.inputs.input_units             = 'secs'
modelspec.inputs.output_units            = 'secs'
modelspec.inputs.time_repetition         = 3.
modelspec.inputs.high_pass_filter_cutoff = 120
```

nodes



Neuroimaging Pypelines

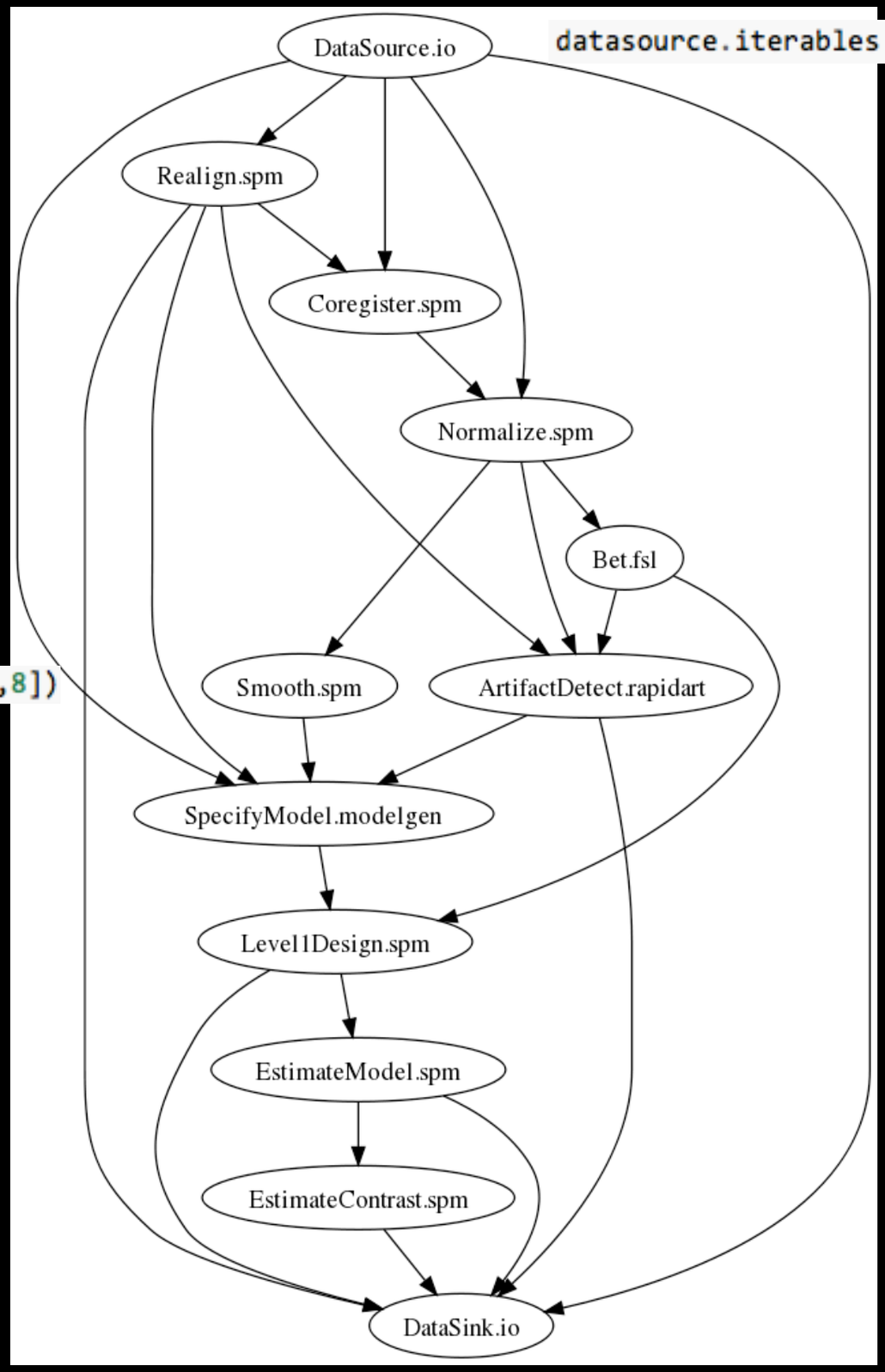
edges

```
l1pipeline.connect([(datasource,realign,[('func','infile')]),
                    # register mean functional to subject surface
                    (datasource,surfregister,[('subject_id','subject_id')]),
                    (realign,surfregister,[('mean_image', 'sourcefile')]),
                    # smooth using freesurfer's mixed-mode smoothing
                    (surfregister, smooth, [('outregfile','regfile')]),
                    (realign, smooth, [('realigned_files', 'sourcefile')]),
                    # generate a mask
                    (smooth,skullstrip,[(('outfile',pickone), 'infile')]),
                    # find outliers
                    (realign,art,[('realignment_parameters','realignment_parameters'),
                                  ('realigned_files','realigned_files')]),
                    (skullstrip,art,[('maskfile','mask_file')]),
                    # design the model
                    (datasource,modelspec,[('subject_id','subject_id')]),
                    (realign,modelspec,[('realignment_parameters','realignment_parameters')]),
                    (smooth,modelspec,[('outfile','functional_runs')]),
                    (art,modelspec,[('outlier_files','outlier_files')]),
                    # generate the SPM design matrix
                    (modelspec,level1design,[('session_info','session_info')]),
                    (skullstrip,level1design,[('maskfile','mask_image')]),
                    # estimate the model parameters
                    (level1design,level1estimate,[('spm_mat_file','spm_design_file')]),
                    # evaluate the contrasts
                    (level1estimate,contrastestimate,[('spm_mat_file','spm_mat_file'),
                                                      ('beta_images','beta_images'),
                                                      ('residual_image','residual_image'),
                                                      ('RPVimage','RPVimage')]),
                  ])
```

Neuroimaging Pypelines

the pipeline

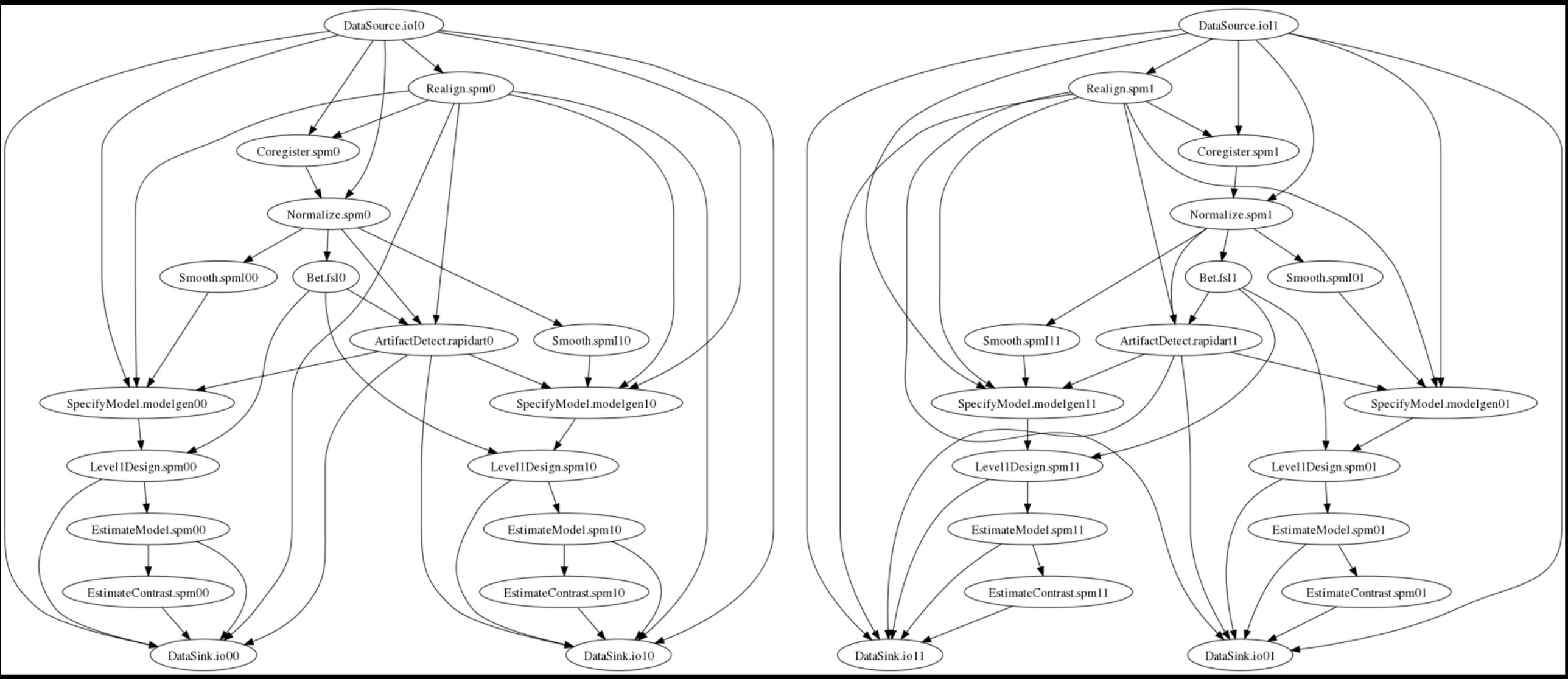
```
datasource.iterables = dict(subject_id=lambda:subject_list)
```




```
smooth.iterables = dict(fwhm=lambda:[4,8])
```

Neuroimaging Pypelines

the pipeline



Neuroimaging Pypelines

challenges

Neuroimaging Pypelines | challenges

Interoperability

- Translate data formats. Ideally no conversion node
- Easily change interfaces with similar functionality

Neuroimaging Pypelines | challenges

Interoperability

- Translate data formats. Ideally no conversion node
- Easily change interfaces with similar functionality

Usability

- There are still python components in scripts
- Syntax needs to be semantically cleaner
- A more visual interface is needed

Neuroimaging Pypelines | challenges

Interoperability	- Translate data formats. Ideally no conversion node - Easily change interfaces with similar functionality
Usability	- There are still python components in scripts - Syntax needs to be semantically cleaner - A more visual interface is needed
Debugging	- What happens when things break? - Are we responsible for FSL support? - How can we help users ask the right questions

Neuroimaging Pypelines | challenges

Interoperability	- Translate data formats. Ideally no conversion node - Easily change interfaces with similar functionality
Usability	- There are still python components in scripts - Syntax needs to be semantically cleaner - A more visual interface is needed
Debugging	- What happens when things break? - Are we responsible for FSL support? - How can we help users ask the right questions
Documentation	- Using Sphinx is a good start - How do maintain it as external executables change?

Neuroimaging Pypelines | challenges

Interoperability	- Translate data formats. Ideally no conversion node - Easily change interfaces with similar functionality
Usability	- There are still python components in scripts - Syntax needs to be semantically cleaner - A more visual interface is needed
Debugging	- What happens when things break? - Are we responsible for FSL support? - How can we help users ask the right questions
Documentation	- Using Sphinx is a good start - How do maintain it as external executables change?
Maintenance	- How do we maintain available interfaces? - How do we handle deprecation and addition?

Neuroimaging Pypelines | challenges

Interoperability	- Translate data formats. Ideally no conversion node - Easily change interfaces with similar functionality
Usability	- There are still python components in scripts - Syntax needs to be semantically cleaner - A more visual interface is needed
Debugging	- What happens when things break? - Are we responsible for FSL support? - How can we help users ask the right questions
Documentation	- Using Sphinx is a good start - How do maintain it as external executables change?
Maintenance	- How do we maintain available interfaces? - How do we handle deprecation and addition?
Dependence	- Reliance on third party software - What if they change delivery platform?

Neuroimaging Pypelines

future

technical

- More interfaces
- XNAT compatibility through pyxnat
- Traited/trait-like input checking
- Web interface
- Amazon web-services compatibility

social

- Publications adding nipype scripts/ graphs as supplementary material
- Collaborative development
- Self-sustaining

Neuroimaging Pypelines

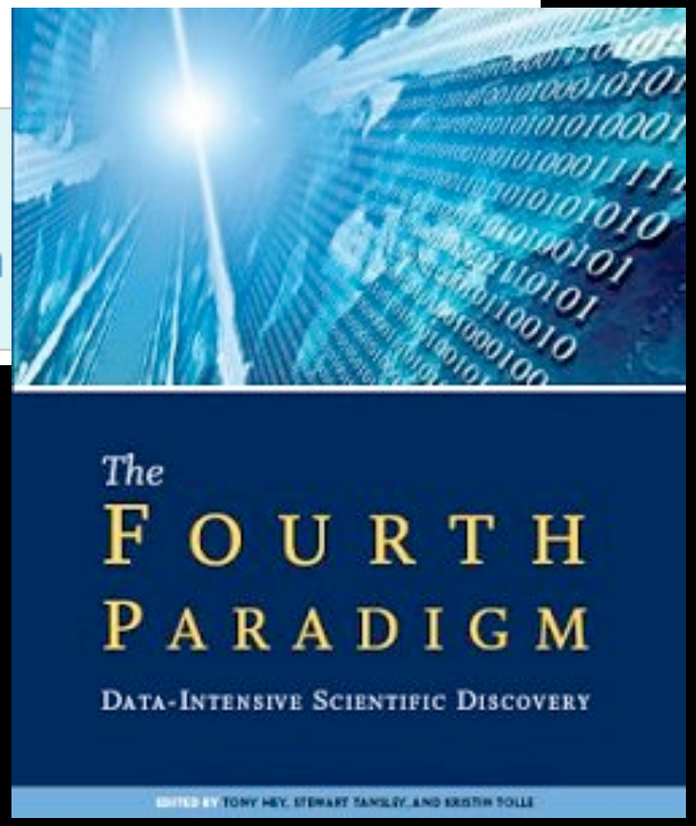
Thanks for the great link...I think one of the things Satra has been pointing out that we need is:

> "The design suite provides a visual scripting application for authoring and sharing workflows and preparing the components that are to be incorporated as executable steps. The aim is to shield the author from the complexities of the underlying applications and enable the author to design and understand workflows without recourse to commissioning specialist and specific applications or hiring software engineers."

I think this is a good goal for the project, but we need to stabilize the infrastructure before we jump on this...
In the short term, maybe what we can shoot for is:

> "Hence there is significant benefit in establishing shared collections of workflows that contain standard processing pipelines for immediate reuse or for repurposing in whole or in part. These aggregations of expertise and resources can help propagate techniques and best practices"

email from cindee madison, nipy(pe) developer



http://research.microsoft.com/en-us/collaboration/fourthparadigm/

Neuroimaging Pypelines | summary

- Generalizable framework for neuroimaging analysis
- Integrates several neuroimaging software
- Python-based (easy to learn, easy to use)
- Pedagogical (you should know what was done)
- Engages the community (opensource)