Brain Connectivity Summer School, Quebec, 15-16 May 2017

Welcome to NeuroPype

Dmitrii Altukhov, Karim Jerbi, David Meunier and Annalisa Pascarella

May 16th, 2017

NeuroPype

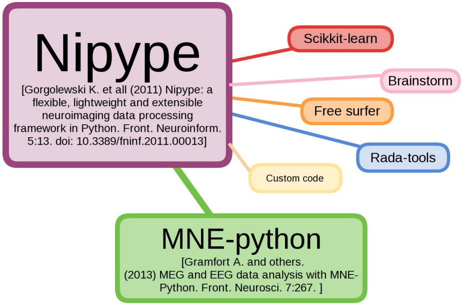
NeuroPype is an **open-source** multi-modal brain data analysis kit which provides **Python-based pipelines** for advanced **multi-thread processing** of fMRI, MEG and EEG data, with a focus on **connectivity and graph analyses**.

NeuroPype is based on **Nipype** framework, a tool developed in fMRI field, which facilitates data analyses by wrapping many commonly-used neuro-imaging software into a **common python framework**

NeuroPype packages

NeuroPype includes three different packages:

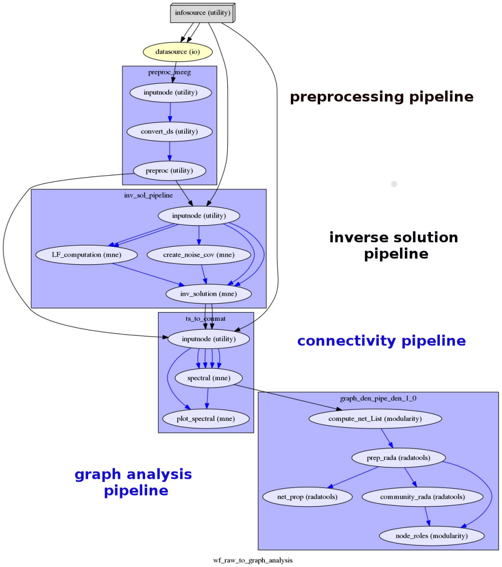
- **neuropype_ephy** includes pipelines for **electrophysiology data analysis**
- **neuropype_graph** allows to study **functional connectivity** exploiting **graph-theoretical metrics** including also **modular partitions**
- **neuropype_cli** is a command line interface for neuropype_ephy package


Nipype
[Gorgolewski K. et all (2011) Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in Python. Front. Neuroinform. 5:13. doi: 10.3389/fninf.2011.00013]
Scikkit-learn
Brainstorm
Free surfer
Rada-tools
Custom code
MNE-python
[Gramfort A. and others. (2013) MEG and EEG data analysis with MNE-Python. Front. Neurosci. 7:267.]

NeuroPype: from raw MEG/EEG to graph properties

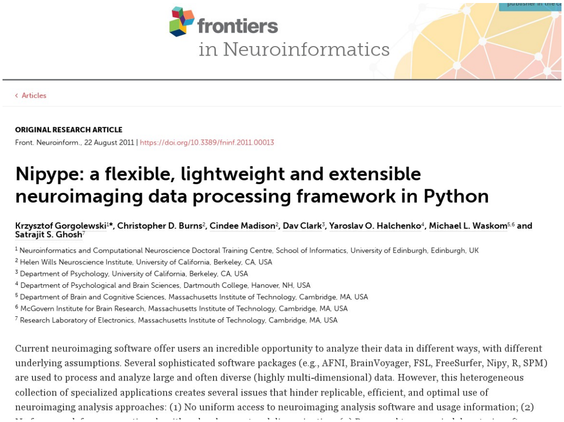
- NeuroPype provides a very common and fast framework to develop **work-flows** for advanced data analyses
- Each **pipeline** could be used stand-alone or as **lego** of a bigger workflow
- Pipelines are defined by **nodes**
  - wrapping of existing software
  - wrapping of function defined by the user

Workflow example



From MEG raw data to spectral connectivity and graph theoretical analysis in source space

Nipype



frontiers in Neuroinformatics

‹ Articles

ORIGINAL RESEARCH ARTICLE

Front. Neuroinform., 22 August 2011 | https://doi.org/10.3389/fninf.2011.00013

Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in Python

Krzysztof Gorgolewski[1]*, Christopher D. Burns[2], Cindee Madison[2], Dav Clark[3], Yaroslav O. Halchenko[4], Michael L. Waskom[5,6] and Satrajit S. Ghosh[7]

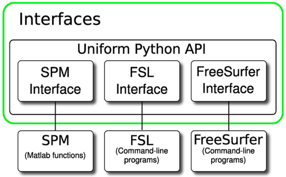
[1] Neuroinformatics and Computational Neuroscience Doctoral Training Centre, School of Informatics, University of Edinburgh, Edinburgh, UK
[2] Helen Wills Neuroscience Institute, University of California, Berkeley, CA, USA
[3] Department of Psychology, University of California, Berkeley, CA, USA
[4] Department of Psychological and Brain Sciences, Dartmouth College, Hanover, NH, USA
[5] Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA, USA
[6] McGovern Institute for Brain Research, Massachusetts Institute of Technology, Cambridge, MA, USA
[7] Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA, USA

Current neuroimaging software offer users an incredible opportunity to analyze their data in different ways, with different underlying assumptions. Several sophisticated software packages (e.g., AFNI, BrainVoyager, FSL, FreeSurfer, Nipy, R, SPM) are used to process and analyze large and often diverse (highly multi-dimensional) data. However, this heterogeneous collection of specialized applications creates several issues that hinder replicable, efficient, and optimal use of neuroimaging analysis approaches: (1) No uniform access to neuroimaging analysis software and usage information; (2)

Nipype

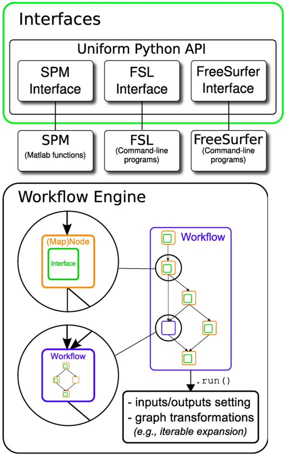
- The workflow design is based on the **Nipype framework**
- Nipype (Neuroimaging in Python: Pipelines and Interfaces) is an open-source, community developed, **Python based** software package that easily interfaces with existing software for efficient analysis of neuroimaging data and **rapid comparative development of algorithms**.
- Nipype provides **Interfaces** to existing neuroimaging software with uniform semantics and facilitates interactions between these packages using **Workflow**

Nipype: 3 main components



Interfaces to external tools that provide a unified way for setting inputs, executing and retrieving outputs

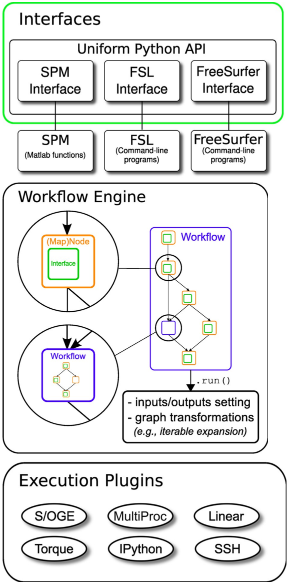
Nipype: 3 main components



Interfaces to external tools that provide a unified way for setting inputs, executing and retrieving outputs

a **workflow engine** that allows **to create analysis pipelines** by connecting inputs and outputs of interfaces as a directed acyclic graph (DAG)

Nipype: 3 main components

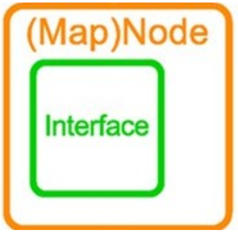


Interfaces **to external tools** that provide a unified way for setting inputs, executing and retrieving outputs

a **workflow engine** that allows **to create analysis pipelines** by connecting inputs and outputs of interfaces as a directed acyclic graph (DAG)

plug-ins that **execute workflows** either locally or in a distributed processing environment

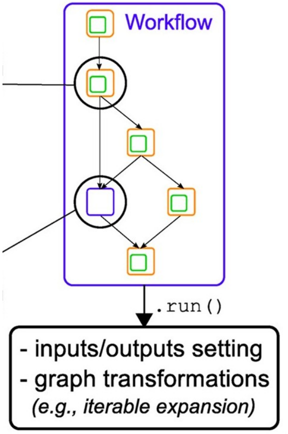
Interfaces and Nodes



To be used in a Workflow the Interfaces are encapsulated in **Node objects**

- each Node is characterized by its unique name and has a least one input and one output
- the Nodes execute the underlying Interface in their own **uniquely named directories**
  - mechanism to isolate and track the outputs resulting from the Interface execution

Workflow



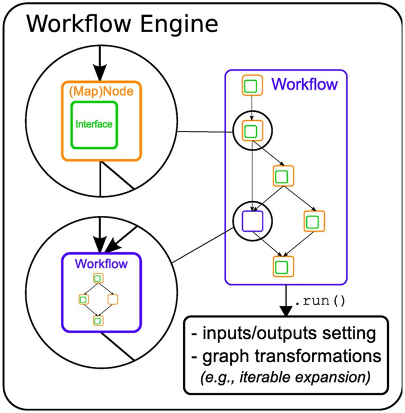
Interfaces encapsulated into Nodes can be connected together within a **Workflow**

- by connecting the outputs of some Node to inputs of another one, the user implicitly specifies dependencies
- to run the Workflow one simply call its **.run()** method

The **workflow engine**

- validates that each Node has a unique name
- checks if all mandatory inputs are set
- cheks if all dependencies are satisfied

Workflow and Nodes



- Node provides also a easy way to implement functions defined by the user
- **Workflow** themselves **can be a Node** of the Workflow graph

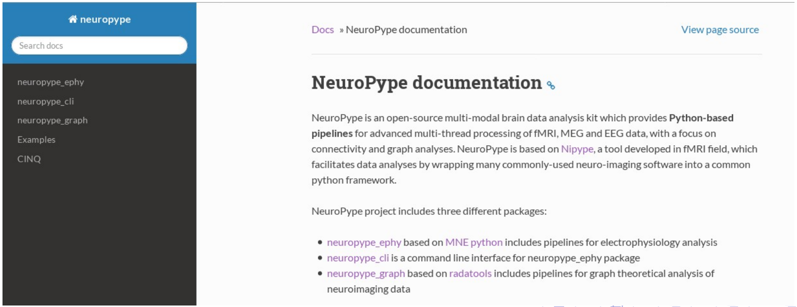
Summary

Main steps to **develop a workflow in nipype framework**

- create a **Workflow** object
- create **Nodes** object
- define the **connections** betwenn the Nodes
- **run** the Workflow object!

Neuropype Packages

NeuroPype online tutorial



NeuroPype documentation

NeuroPype is an open-source multi-modal brain data analysis kit which provides **Python-based pipelines** for advanced multi-thread processing of fMRI, MEG and EEG data, with a focus on connectivity and graph analyses. NeuroPype is based on Nipype, a tool developed in fMRI field, which facilitates data analyses by wrapping many commonly-used neuro-imaging software into a common python framework.

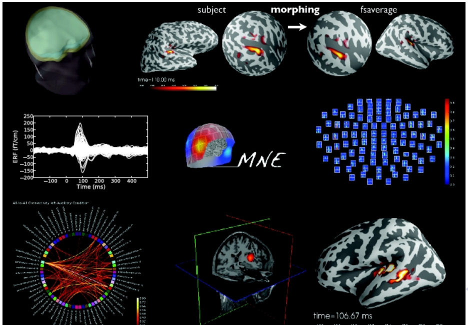
NeuroPype project includes three different packages:

- neuropype_ephy based on MNE python includes pipelines for electrophysiology analysis
- neuropype_cli is a command line interface for neuropype_ephy package
- neuropype_graph based on radatools includes pipelines for graph theoretical analysis of neuroimaging data

NeuroPype-ephy

NeuroPype-ephy is a package based on **MNE-python** software http://martinos.org/mne/ and includes pipelines for **electrophysiology** data analysis. Current implementations allow for

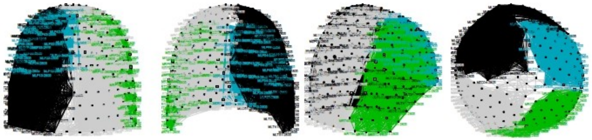
- MEG/EEG data import
- MEG/EEG data pre-processing and cleaning by an automatic removal of eyes and heart related artefacts
- sensor or **source-level** connectivity analyses

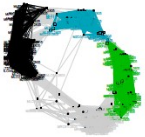


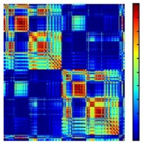
NeuroPype-graph

NeuroPype-graph is a package based on **radatools** software http://deim.urv.cat/ sergio.gomez/radatools.php and includes pipelines for **graph theoretical analysis of neuroimaging data**. Current implementations allow to construct pipelines

- from nifti 4D (after preprocessing) to connectivity matrices
- from connectivity matrices to graph analysis
- from integer matrices (normally coclassification matrices) to graph analysis

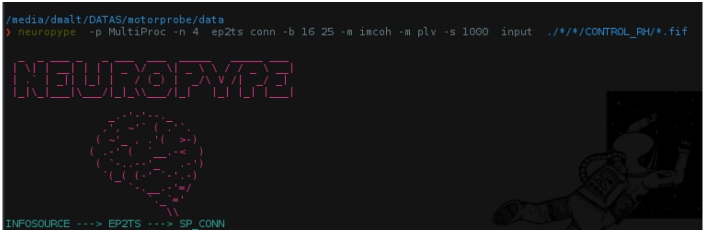






NeuroPype-cli

Command line interface for neuropype_ephy



```
/media/dmalt/DATAS/motorprobe/data
❯ neuropype  -p MultiProc -n 4  ep2ts conn -b 16 25 -m imcoh -m plv -s 1000  input  ./*/*/CONTROL_RH/*.fif

INFOSOURCE ---> EP2TS ---> SP_CONN
```

Key idea: Use globbing to improve flexibility of the pipeline

- Flexible input
- Connect nodes on the go
- Same pattern for launching remotely
- Works on clusters
- **help** pages for each option and command

NeuroPype doors

Can I use NeuroPype on

NeuroPype doors

Can I use NeuroPype on

- raw MEG, EEG data?

NeuroPype doors

Can I use NeuroPype on

- raw MEG, EEG data? **YES**

NeuroPype doors

Can I use NeuroPype on

- raw MEG, EEG data? **YES**
- time series of data in sensor or source space?

NeuroPype doors

Can I use NeuroPype on

- raw MEG, EEG data? **YES**
- time series of data in sensor or source space? **YES**

NeuroPype doors

Can I use NeuroPype on

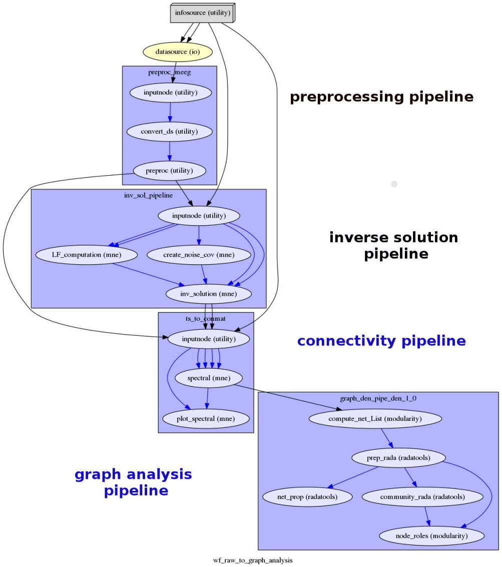
- raw MEG, EEG data? **YES**
- time series of data in sensor or source space? **YES**
- connectivity matrices?

NeuroPype doors

Can I use NeuroPype on

- raw MEG, EEG data? **YES**
- time series of data in sensor or source space? **YES**
- connectivity matrices? **YES**

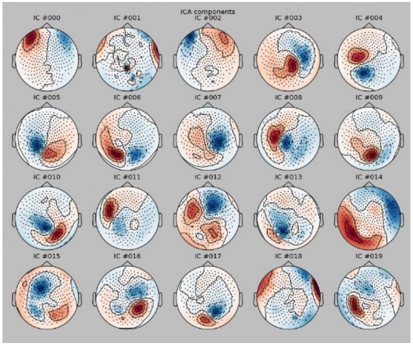
Workflow example



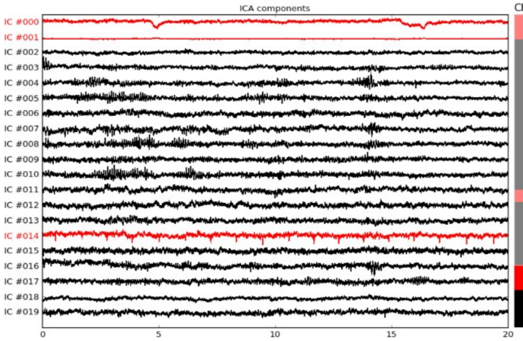
From MEG raw data to spectral connectivity and graph theoretical analysis in source space

Visualization - preprocessing

ICA topographies

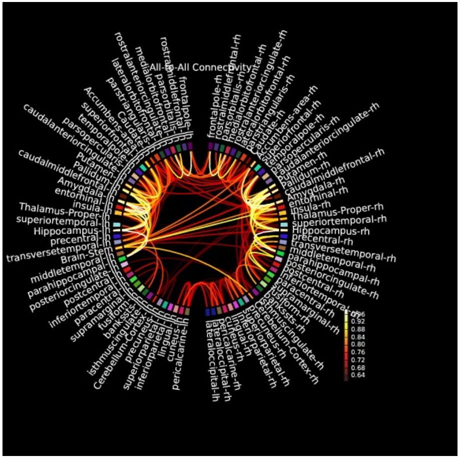


ICA time series

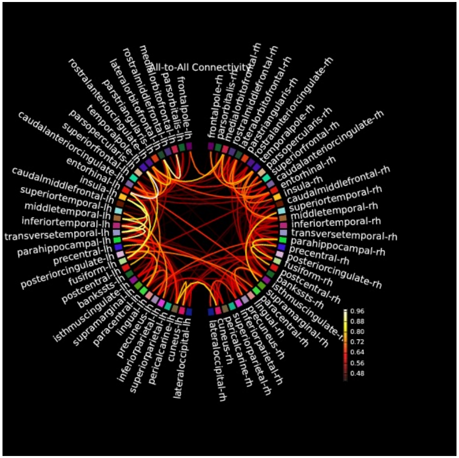


Visualization - connectivity on source space

connectivy on mixed source space

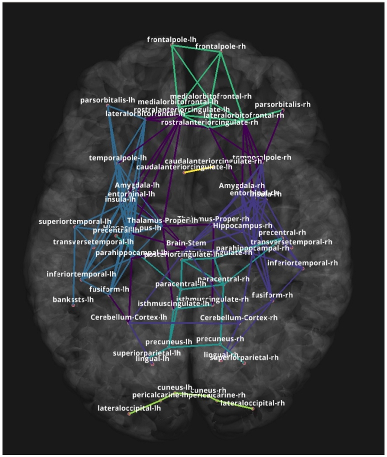


connectivy on cortical surface

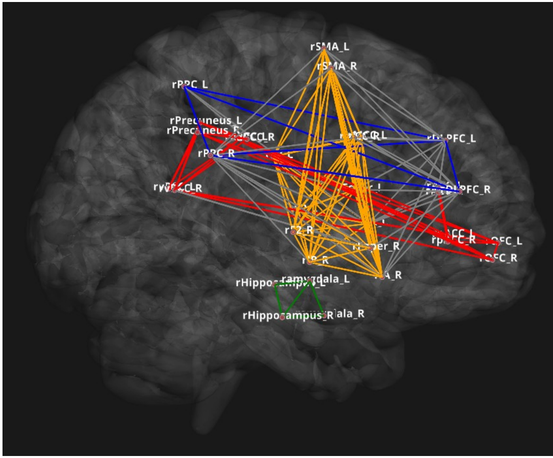


Visualization - modular graphs

graph - MEG data



modular graph - fMRI data



Advantages and strengths

- **Multi-threading**
- **Caching**
- Access to numerous **python-wrappers** for image analysis
- **Multimodal analyses**
- **open-source and free**

NeuroPype Team and collaborators

Karim Jerbi **Dmitrii Altukhov** **David Meunier** **Annalisa Pascarella**

CoCo lab users and collaborators

Reference

Bullmore E, Sporns O (2009) Complex brain networks: graph theoretical analysis of structural and functional systems, Nat Rev Neurosci 10:186-198

Gorgolewski K, Burns CD, Madison C, Clark D, Halchenko YO, Waskom ML, Ghosh SS (2011) Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in Python, Front. Neuroinform. 5:13. doi:10.3389/fninf.2011.00013

Gramfort A, Luessi M, Larson E, Engemann D A, Strohmeier D, Brodbeck C, Parkkonen L and Hämäläinen M (2014), MNE software for processing MEG and EEG data, Neuroimage, 86, 446-460

Gramfort A, Luessi M, Larson E, Engemann D A, Strohmeier D, Brodbeck C, Goj R, Jas M, Brooks T, Parkkonen L and Hämäläinen M (2013), MEG and EEG data analysis with MNE-Python, Frontiers in Neuroscience