Welcome to NeuroPype

Dmitrii Altukhov, Karim Jerbi, David Meunier and Annalisa Pascarella

May 16th, 2017

Université de Montréal

NeuroPype

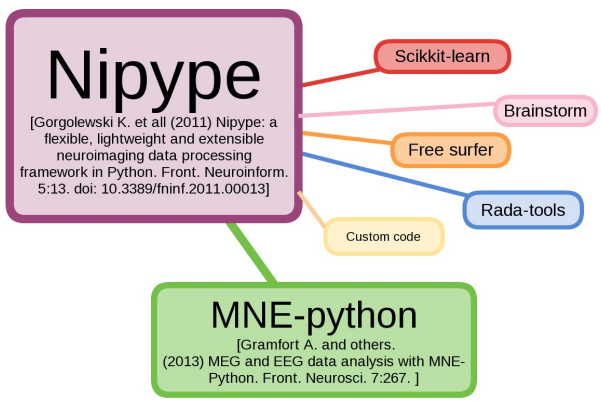
NeuroPype is an **open-source** multi-modal brain data analysis kit which provides **Python-based pipelines** for advanced **multi-thread processing** of fMRI, MEG and EEG data, with a focus on **connectivity and graph analyses**.

- 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

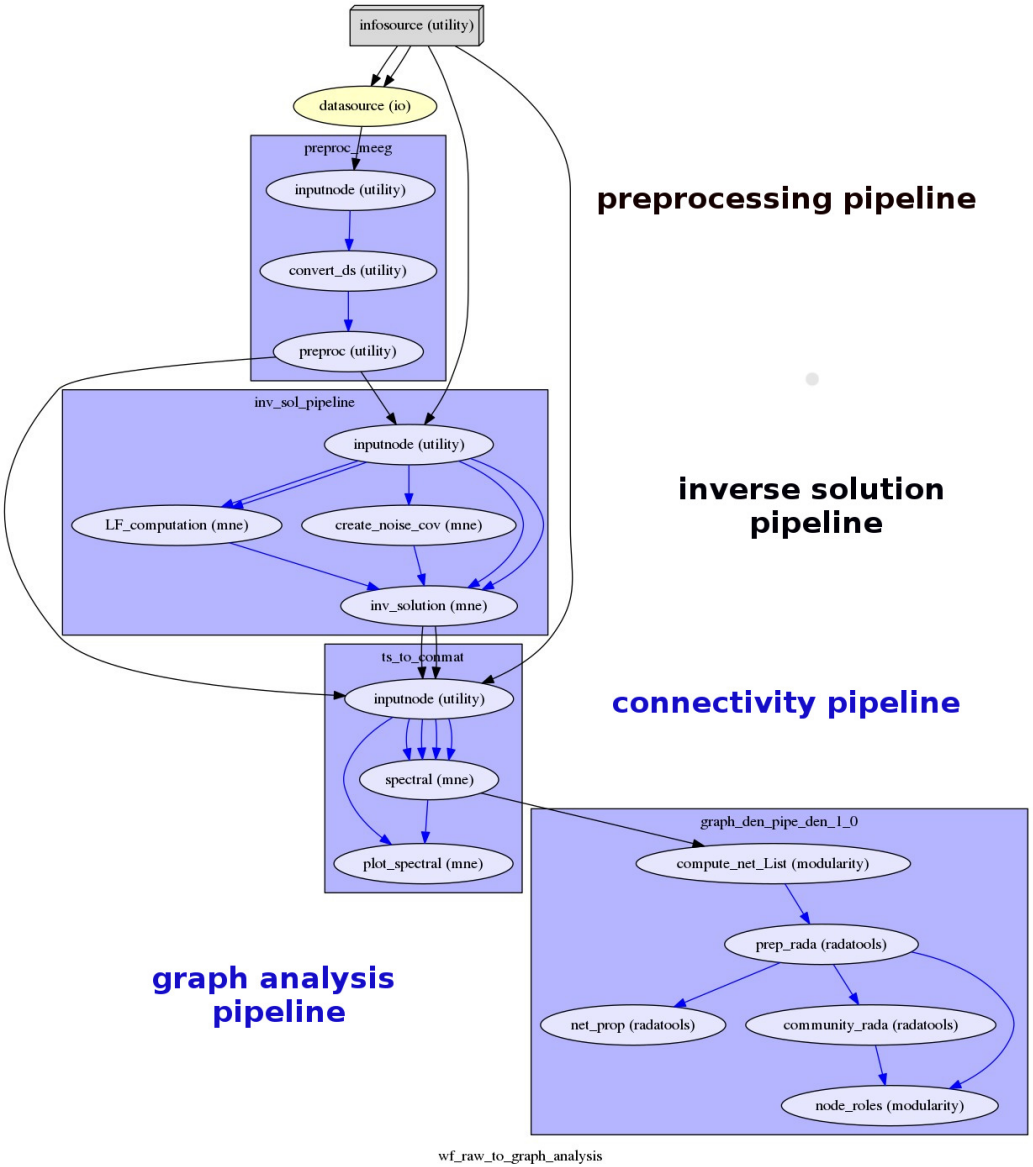
NeuroPype

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**.

- 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
- includes three different packages:
  - **neuropype_ephy** includes pipelines for electrophysiology 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



NeuroPype: from raw MEG/EEG to graph properties



- NeuroPype provides a very common and fast framework to develop **workflows** for advanced data analyses
- Each **pipeline** could be used standalone or as **lego** of a bigger workflow: its output could be the input of another pipeline
- Pipelines are defined by **nodes**, which maybe wrapping of existing software as well as providing easy ways to implement function defined by the user
- This is an example of workflow created by using NeuroPype: from MEG raw data to spectral connectivity and graph theoretical analysis in source space

Nipype

‹ 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

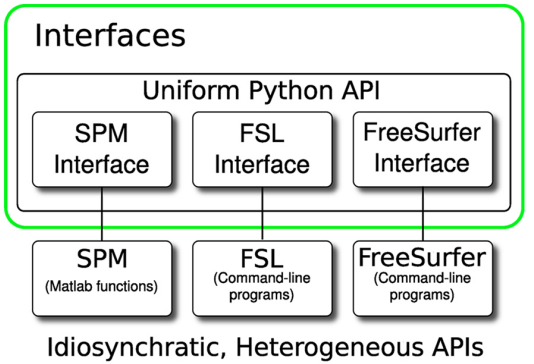
Current neuroimaging software offer users an incredible opportunity to analyze their data in different ways, with different underlying assumptions. Several sophisticated software packages (e.g., AFNI, BrainVoyager, FSL, FreeSurfer, Nipy, R, SPM) are used to process and analyze large and often diverse (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)

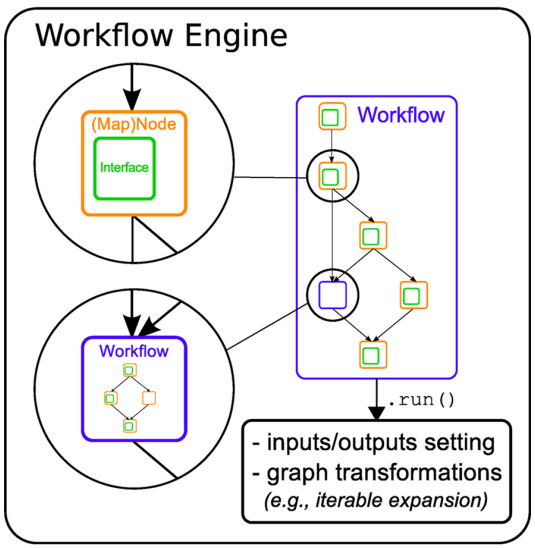
Nipype

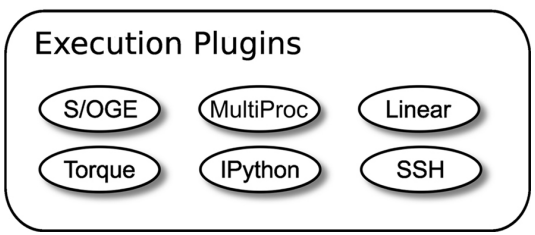
- The workflow design is based on the Nipype framework, allowing in a very readable fashion the design of processing pipelines
- Nipype (Neuroimaging in Python: Pipelines and Interfaces) is an open-source, community developed, **Python based** software package that easily interfaces with existing software 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**
- The workflow execution engine has a plug-in architecture and supports both local execution on multi-core machines and remote execution on clusters

Nipype:
Neuroimaging in Python
Pipelines and Interfaces

Nipype: main components





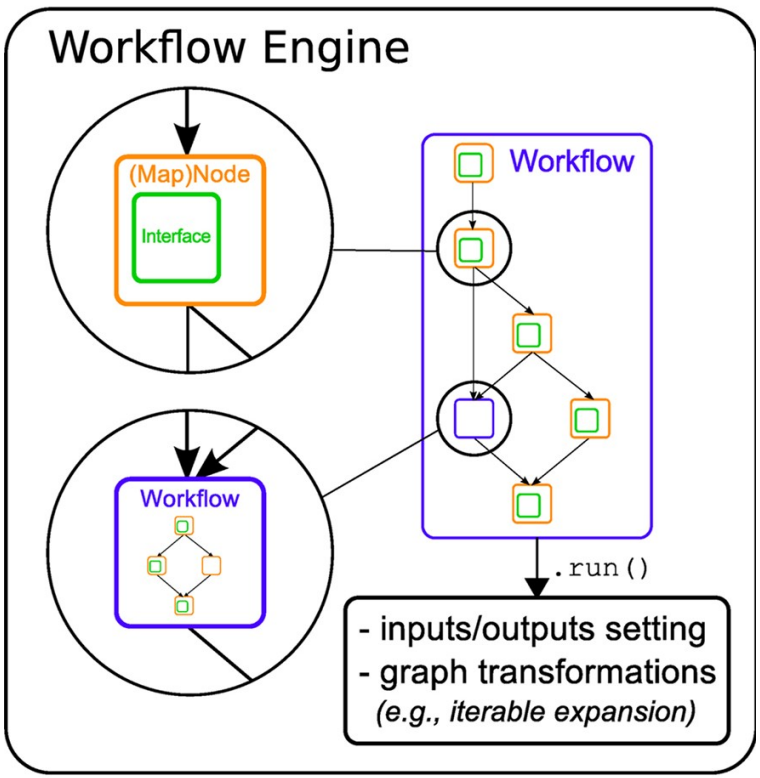


Nipype consists of three main components:

- **Interfaces** to external tools that provide a unified way for setting inputs, executing and retrieving outputs
  - the goal of Interfaces is to provide a uniform mechanism for accessing analysis tool from neuroimaging software packages (e.g. Freesurfer, FSL, SPM, ...)
- 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
  - no changes are needed to the Workflow to switch between these execution modes. The user simply calls the Workflow **run** function with a different plug-in and its arguments

Interfaces and Nodes

Nipype provides a framework for connecting Interfaces to create a **data analysis Workflow**



- In order to be used in a Workflow the Interfaces have to be encapsulated in **Node objects**
  - they execute the underlying Interface in their own **uniquely named directories**, thus providing a mechanism to isolate and track the outputs resulting from the Interface execution
- 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
  - the dependencies in a Workflow are represented internally as a **DAG**
- Workflow themselves can be a Node of the Workflow graph
- Node provides also a easy way to implement functions defined by the user

Scratch code

```
main_workflow = pe.Workflow(name=correl_analysis_name)
main_workflow.base_dir = main_path

infosource = create_infosource()  # info source Node
datasource = create_datasource()  # data source Node

main_workflow.connect(infosource, 'subject_id', datasource, 'subject_id')

create_ts_node = pe.Node(interface=Function(input_names=['raw_fname'],
                                            output_names=['ts_file'],
                                            function=create_ts),
                         name='create_ts')

main_workflow.connect(datasource, 'raw_file', create_ts_node, 'raw_fname')

spectral_workflow = \
    create_pipeline_time_series_to_spectral_connectivity(main_path,
                                                         con_method=con_method)

main_workflow.connect(create_ts_node, 'ts_file', spectral_workflow, 'inputnode.ts_file')

main_workflow.run(plugin='MultiProc', plugin_args={'n_procs': 8})
```

Advantages and strengths

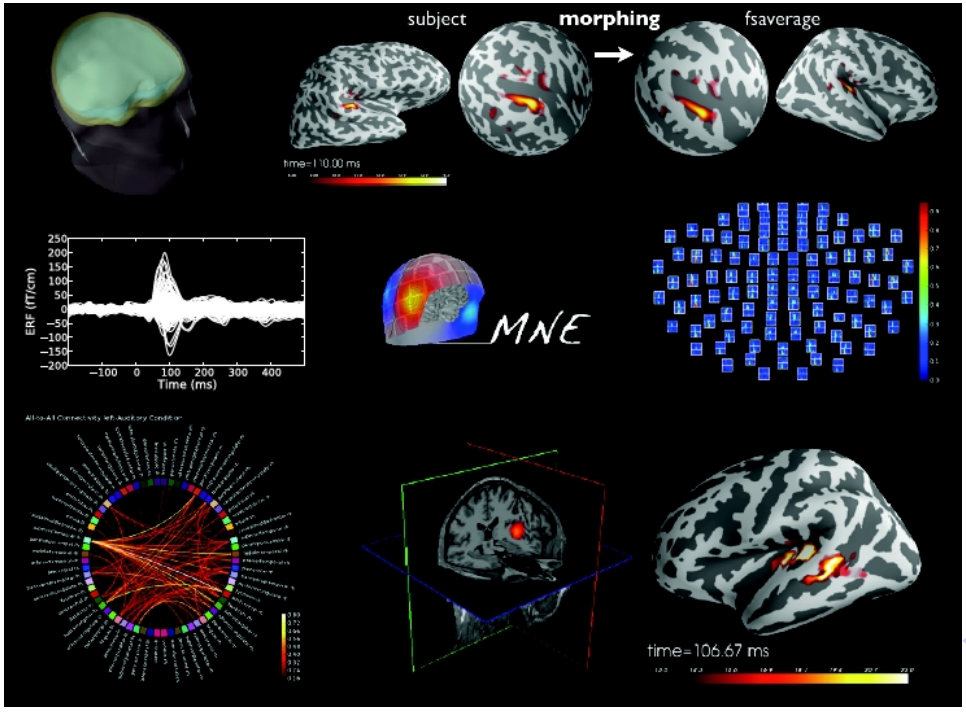
- **Multi-threading**: the implementation of Multi-threading in Nipype is very easy, and can be either made for multi-processing on a same machine with multiple cores (Multiproc plugin) or a cluster with multiple machine in parallel (q-sub/ipython plugin)
- **Caching**: Nipype has a framework allowing the storage of intermediate files, as well as testing if the source code of each node has been modified. Hence, if a part of the pipeline is modified, only the modified parts will be recomputed, having a significant impact on the speed of the analysis as well.
- Access to numerous **python-wrappers** for image analysis: being based on the same framework as Nipype, Neuropype can benefit of all the interfaces already available for neuro-imaging analysis if both MRI and MEG data are available for a set of subjects.
- **Multimodal analyses**: the processing tools for different modalities (e.g.fMRI and MEG data) are all wrapped within the same framework, and can be simultaneously analyzed
- Finally, being written in python, Neuropype is highly readable, and contrarily to other high-level scientific language such as Matlab, is **open-source and free**, allowing for code sharing and easier reproducible results

Neuropype Packages

NeuroPype-ephy

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

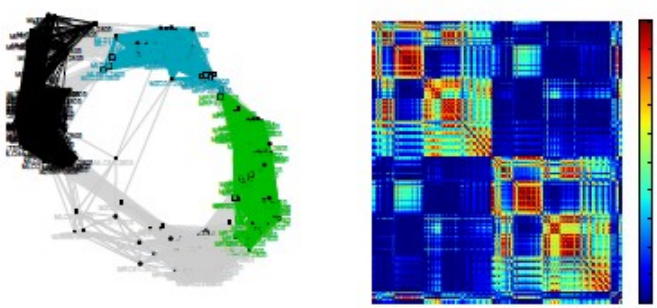
- MEG/EEG data import
- MEG/EEG data pre-processing and cleaning by an automatic removal of eyes and heart related artefacts
- sensor or **source-level** connectivity analyses



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

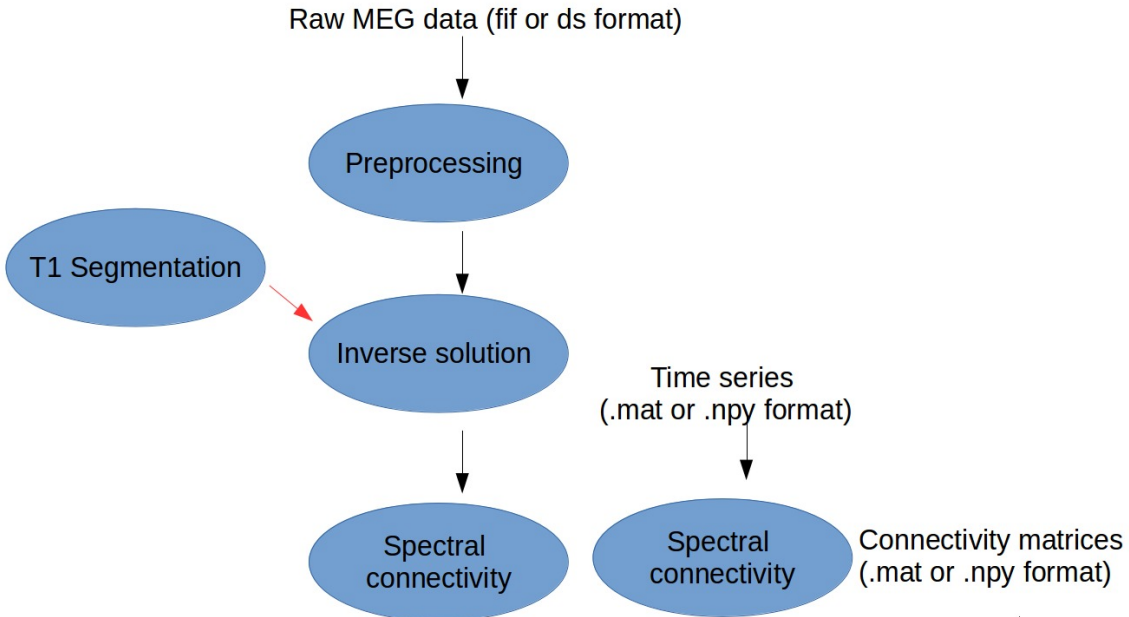
Key idea: Use globbing to improve flexibility of the pipeline

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

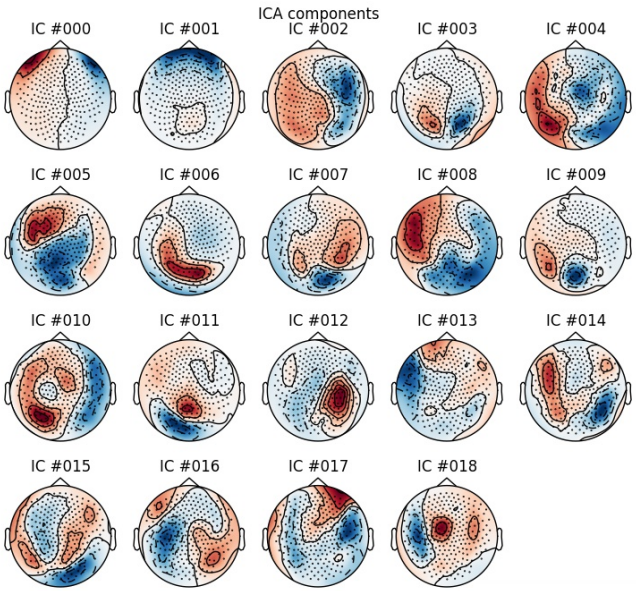
NeuroPype doors

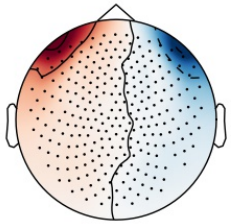
Neuropype packages define a set of different pipelines (also called workflow) that can be used stand-alone or as brick of a bigger workflow: the input of a pipeline will be the output of another pipeline.
For each possible workflow the input data can be specified in three different ways:

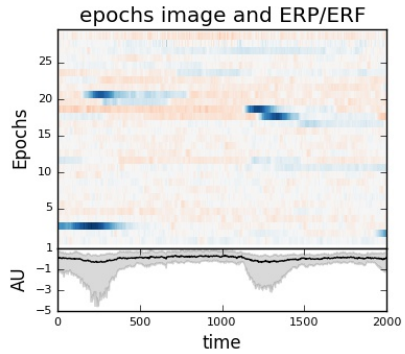
- raw MEG data in .fif and .ds format
- time series of connectivity matrices in .mat (Matlab) or .npy (Numpy) format
- connectivity matrices in .mat (Matlab) or .npy (Numpy) format

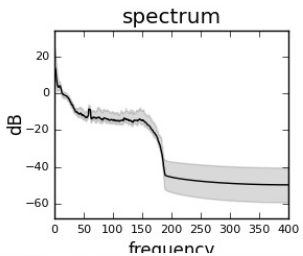


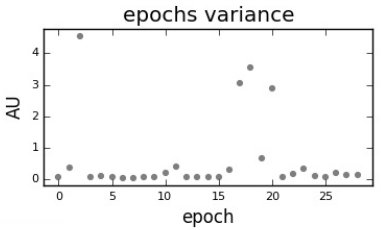
Visualization

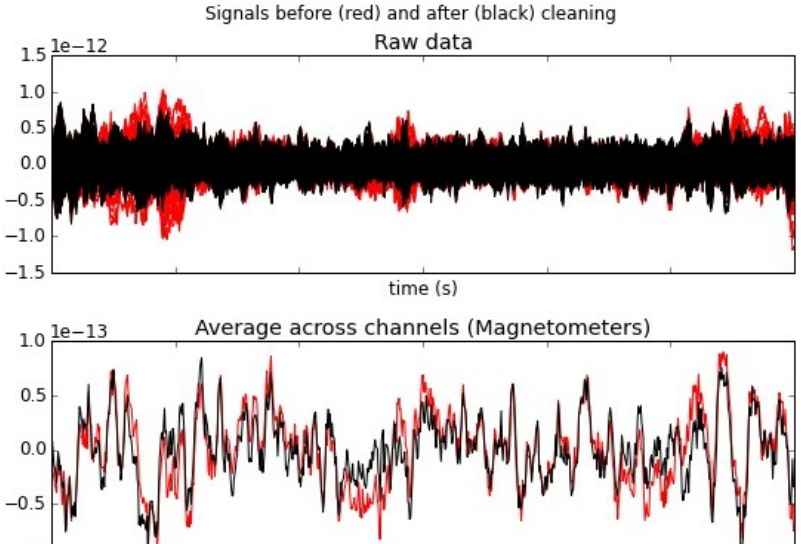




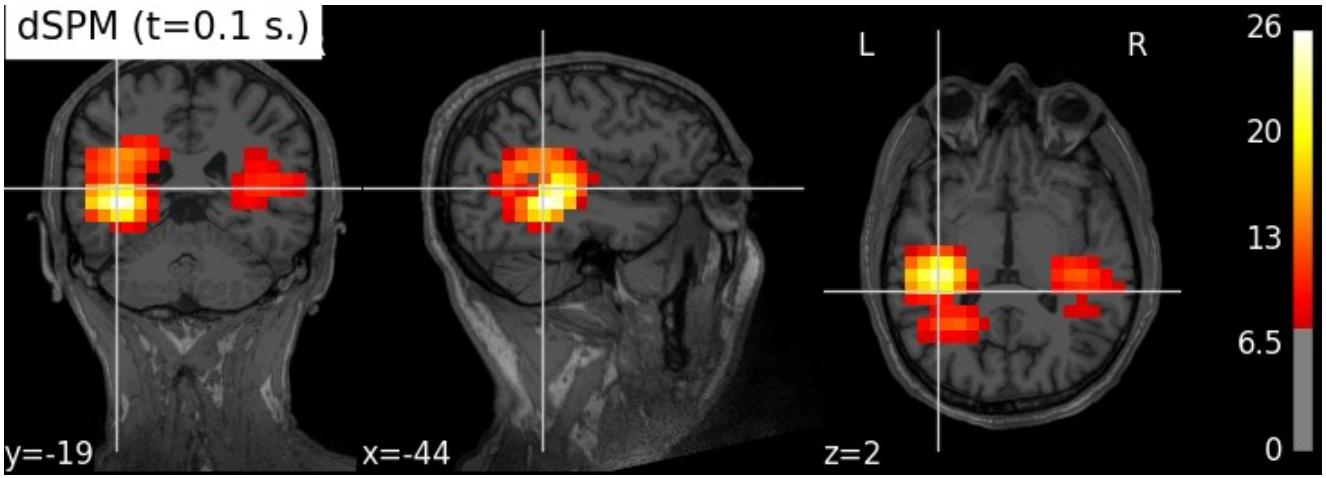


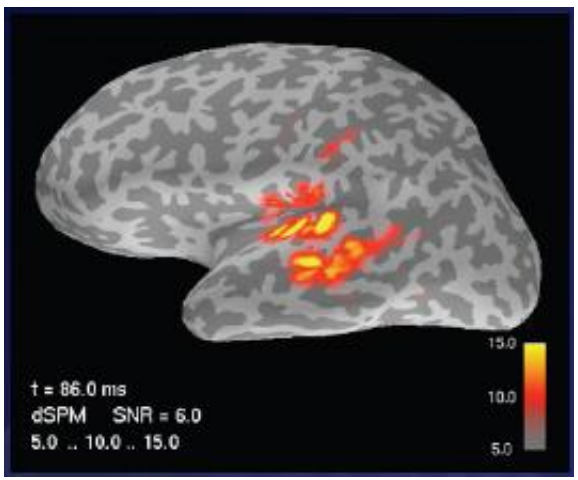


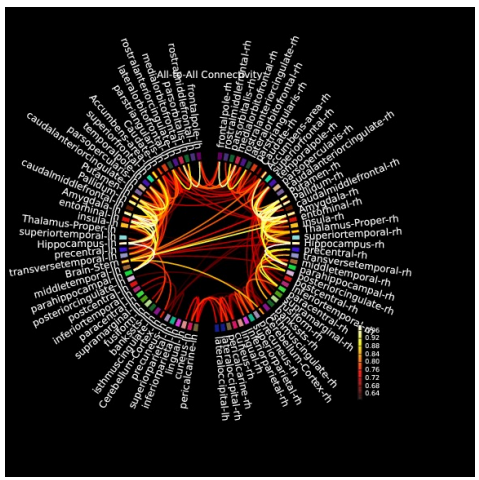


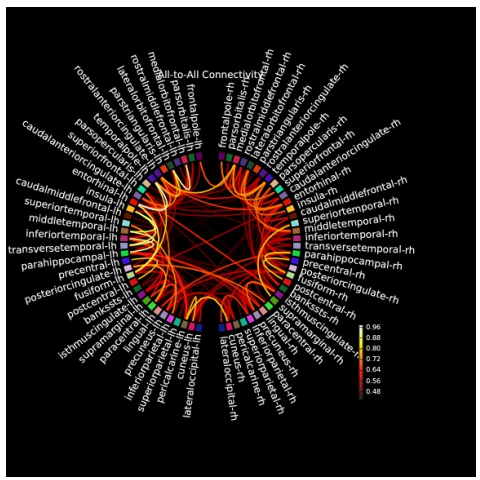


Visualization









NeuroPype Team and collaborators

Karim Jerbi

Dmitrii Altukhov

David Meunier

Annalisa Pascarella

CoCo lab users and collaborators

Lyon People! -¿ put some pictures/names

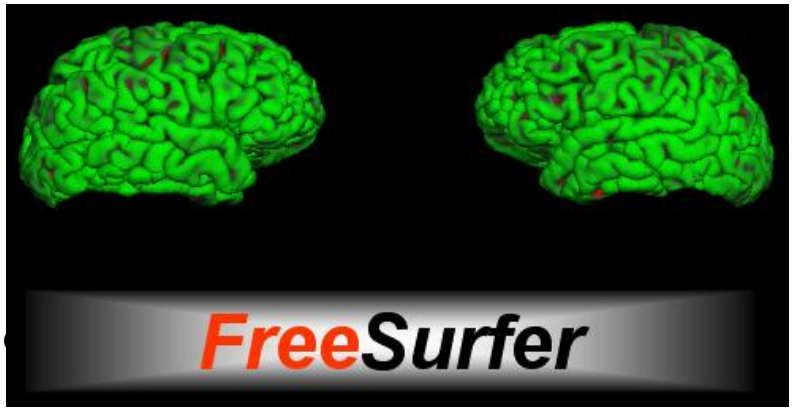
Software

http://FreeSurfer.net/fswiki

http://surfer.nmr.mgh.harvard.edu/

http://surfer.nmr.mgh.harvard.edu/fswiki/recon-all

http://surfer.nmr.mgh.harvard.edu/fswiki/Recommend



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