Command line tools for MEEG data processing

December 20, 2016

Outline

Pipeline for MEEG data analyses

Steps of analyses

1. Preprocessing
   - filtering
   - downsampling
   - artifacts removal
   - epochs creation
   - source reconstruction
2. Features computation
   - PSD
   - Connectivity
   - C1/C1
   - ...
3. Features collection
4. Machine learning, stats, etc..

Pipeline for MEEG data analyses

Steps of analyses

1. Preprocessing
    - filtering
    - downsampling
    - artifacts removal
    - epochs creation
    - source reconstruction
2. Features computation
    - PSD
    - Connectivity
    - C1/C1
    - ...
3. Features collection
4. Machine learning, stats, etc..

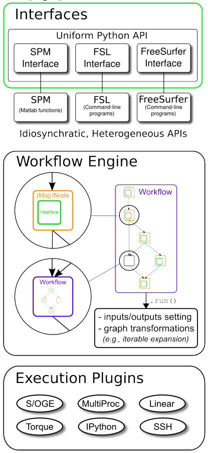
NEUROPYPE:

Nipype:
Neuroimaging in Python
Pipelines and Interfaces

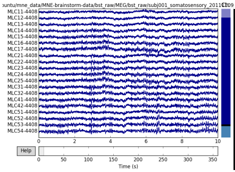
MNE
MEG + EEG ANALYSIS & VISUALIZATION

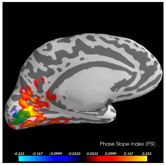
neuropype_ephy package

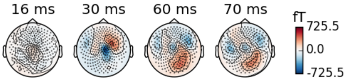
Nipype

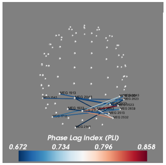


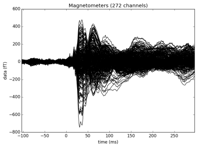
MNE python



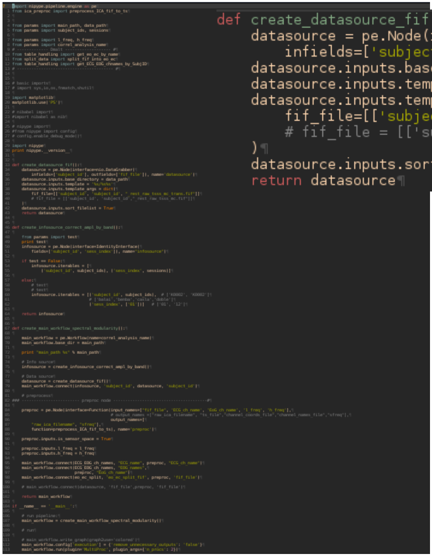




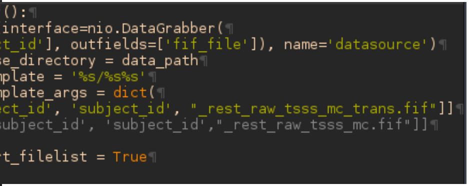




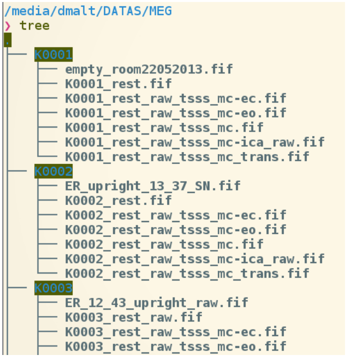
Motivation for command line interface



```python
import nipype.pipeline.engine as pe
from ica_preproc import preprocess_ICA_fif_to_ts


from params import main_path, data_path
from params import subject_ids, sessions

from params import l_freq, h_freq
from params import correl_analysis_name
# ------------- Dmalt ------------------ #
from table_handling import get_eo_ec_by_name
from split_data import split_fif_into_eo_ec
from table_handling import get_ECG_EOG_chnames_by_SubjID
# ---------------------------------------- #


# basic imports
# import sys,io,os,fnmatch,shutil

import matplotlib
matplotlib.use('PS')

# nibabel import
#import nibabel as nib

# nipype import
#from nipype import config
# config.enable_debug_mode()

import nipype
print nipype.__version__


def create_datasource_fif():
    datasource = pe.Node(interface=nio.DataGrabber(
        infields=['subject_id'], outfields=['fif_file']), name='datasource')
    datasource.inputs.base_directory = data_path
    datasource.inputs.template = '%s/%s%s'
    datasource.inputs.template_args = dict(
        fif_file=[['subject_id', 'subject_id', "_rest_raw_tsss_mc_trans.fif"]]
        # fif_file = [['subject_id', 'subject_id',"_rest_raw_tsss_mc.fif"]]
    )
    datasource.inputs.sort_filelist = True
    return datasource


def create_infosource_correct_ampl_by_band():

    from params import test
    print test
    infosource = pe.Node(interface=IdentityInterface(
        fields=['subject_id', 'sess_index']), name="infosource")

    if test == False:
        infosource.iterables = [
            ('subject_id', subject_ids), ('sess_index', sessions)]

    else:
        # test
        # test
        infosource.iterables = [('subject_id', subject_ids),  # ['K0002', 'K0002']
                                # ['balai','benba','casla','doble']
                                ('sess_index', ['01'])]   # ['01', '12']

    return infosource


def create_main_workflow_spectral_modularity():

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

    print "main_path %s" % main_path

    # Info source
    infosource = create_infosource_correct_ampl_by_band()

    # Data source
    datasource = create_datasource_fif()
    main_workflow.connect(infosource, 'subject_id', datasource, 'subject_id')

    # preprocess
### ------------------------ preproc node ------------------------------------#

    preproc = pe.Node(interface=Function(input_names=["fif_file", 'ECG_ch_name', 'EoG_ch_name', 'l_freq', 'h_freq'],
                                         # output_names =["raw_ica_filename", "ts_file","channel_coords_file","channel_names_file","sfreq"],
                                         output_names=[
        "raw_ica_filename", "sfreq"],
        function=preprocess_ICA_fif_to_ts), name='preproc')

    preproc.inputs.is_sensor_space = True

    preproc.inputs.l_freq = l_freq
    preproc.inputs.h_freq = h_freq

    main_workflow.connect(ECG_EOG_ch_names, "ECG_name", preproc, "ECG_ch_name")
    main_workflow.connect(ECG_EOG_ch_names, "EOG_names",
                          preproc, "EoG_ch_name")
    main_workflow.connect(eo_ec_split, 'eo_ec_split_fif', preproc, 'fif_file')

    # main_workflow.connect(datasource, 'fif_file',preproc, 'fif_file')

    return main_workflow

if __name__ == '__main__':

    # run pipeline:
    main_workflow = create_main_workflow_spectral_modularity()

    # run

    # main_workflow.write_graph(graph2use='colored')
    main_workflow.config['execution'] = {'remove_unnecessary_outputs': 'false'}
    main_workflow.run(plugin='MultiProc', plugin_args={'n_procs': 2})
```



```python
def create_datasource_fif():
    datasource = pe.Node(interface=nio.DataGrabber(
        infields=['subject_id'], outfields=['fif_file']), name='datasource')
    datasource.inputs.base_directory = data_path
    datasource.inputs.template = '%s/%s%s'
    datasource.inputs.template_args = dict(
        fif_file=[['subject_id', 'subject_id', "_rest_raw_tsss_mc_trans.fif"]]
        # fif_file = [['subject_id', 'subject_id',"_rest_raw_tsss_mc.fif"]]
    )
    datasource.inputs.sort_filelist = True
    return datasource
```



```
/media/dmalt/DATAS/MEG
❯ tree
.
├── K0001
│   ├── empty_room22052013.fif
│   ├── K0001_rest.fif
│   ├── K0001_rest_raw_tsss_mc-ec.fif
│   ├── K0001_rest_raw_tsss_mc-eo.fif
│   ├── K0001_rest_raw_tsss_mc.fif
│   ├── K0001_rest_raw_tsss_mc-ica_raw.fif
│   └── K0001_rest_raw_tsss_mc_trans.fif
├── K0002
│   ├── ER_upright_13_37_SN.fif
│   ├── K0002_rest.fif
│   ├── K0002_rest_raw_tsss_mc-ec.fif
│   ├── K0002_rest_raw_tsss_mc-eo.fif
│   ├── K0002_rest_raw_tsss_mc.fif
│   ├── K0002_rest_raw_tsss_mc-ica_raw.fif
│   └── K0002_rest_raw_tsss_mc_trans.fif
├── K0003
│   ├── ER_12_43_upright_raw.fif
│   ├── K0003_rest_raw.fif
│   ├── K0003_rest_raw_tsss_mc-ec.fif
│   ├── K0003_rest_raw_tsss_mc-eo.fif
```

Solving this problem with Unix command line

Globbing

- * - matches any quantity or character
- ? - matches one occurence of any character
- \X - escape special character (lets you use * or ? in matching)
- [x-y] - matches numbers in range from x to y
- {A,B,...} - matches either A, B or any other string in curly braces

▸ Wildcards tutorial

In most cases we need only the '*' symbol

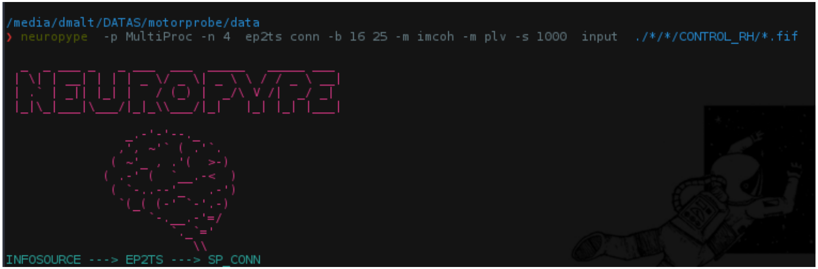
Examples:

```
# List all files in the current directory
$ ls *
# List all .fif files in dataset subdirectories
$ ls ./*/*.fif
```

Installation

```
$ git clone https://github.com/dmalt/neuropype_cli.git
$ cd neuropype_cli
$ pip install –editable .
```

Command line interface for neuropype



```
/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
```

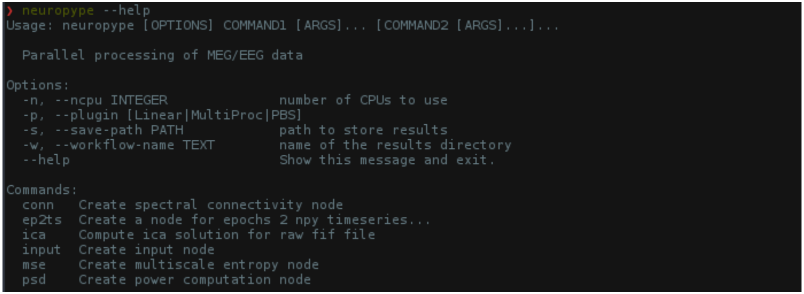
```
INFOSOURCE ---> EP2TS ---> SP_CONN
```

Key idea: Use globbing to improve flexibility of the pipeline

Features

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

Syntax



```
❯ neuropype --help
Usage: neuropype [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...

  Parallel processing of MEG/EEG data

Options:
  -n, --ncpu INTEGER              number of CPUs to use
  -p, --plugin [Linear|MultiProc|PBS]
  -s, --save-path PATH            path to store results
  -w, --workflow-name TEXT        name of the results directory
  --help                          Show this message and exit.

Commands:
  conn   Create spectral connectivity node
  ep2ts  Create a node for epochs 2 npy timeseries...
  ica    Compute ica solution for raw fif file
  input  Create input node
  mse    Create multiscale entropy node
  psd    Create power computation node
```

Workflow options

- -p, –plugin - controls parallel execution of the workflow
  Possible values:
  Linear - serial execution of pipeline
  MultiProc - parallel execution on local machine
  PBS - parallel execution on cluster
- -n - number of processors to use;
  Valid only for parallel plugins (MultiProc, PBS)
- -w, –workflow-name - name of the workflow.
  Output of the pipeline will be stored in a folder with this name
- -s, –save-path - path to a folder where we want to store results
- –help - show help

Output folder structure

- workflow_name
    - __keys__Subj1_cond1__epochs-epo_fif
        - con_method_imcoh_freq_band_15.0.30.0
        - con_method_pli_freq_band_8.0.12.0
        - ep2ts
        - mse
        - pwr
        - path_node
    - __keys__Subj1_cond2__epochs-epo_fif
        - ...
    - __keys__Subj2_cond1__epochs-epo_fif
        - ...
    - ...

Supported nodes (to be improved)

- input - pipeline entry point.
  INPUT: list of relative paths to files for processing (use globbing here)

  OUTPUT: same as INPUT
- ep2ts - convert -epo.fif file with epochs to numpy format
  INPUT: epochs in .fif format

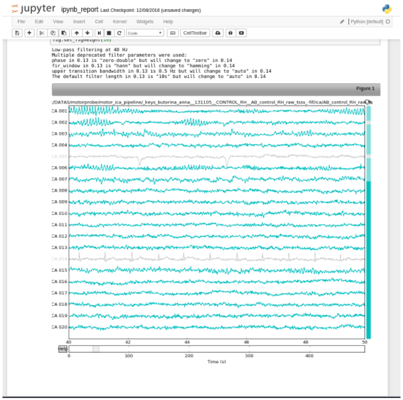
  OUTPUT: .npy file with epochs
- ica - compute ICA solution
  INPUT: list of raw files

  OUTPUT: .html report, ICA solution in .fif, ICA timeseries in .fif, copy of raw file to apply ICA in .fif
- mse - compute multiscale entropy
  INPUT: list of epochs in .npy format (n_trials x n_sensors x n_times matrices)

  OUTPUT: .npz file (same as .npy but for several variables) with two arrays: std_mse and mean_mse
- psd - compute power spectral density
  INPUT: files with epochs in .fif

  OUTPUT: .npz file with power spectral density matrices for each epoch (psds) and frequencies (freqs)
- conn - compute connectivity
  INPUT: epoch files in .npy format
  OUTPUT: connectivity matrices in .npy format

ICA preprocessing

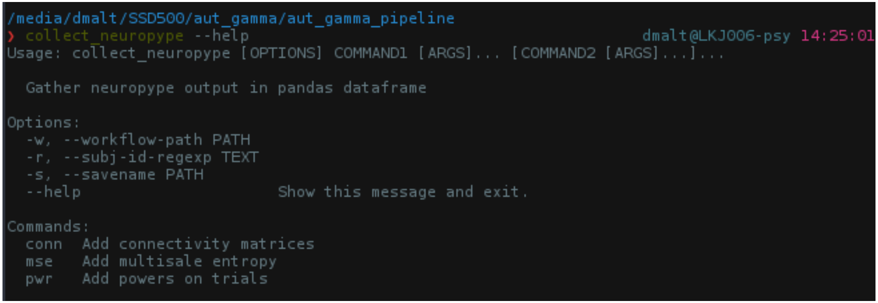


```
Low-pass filtering at 40 Hz
Multiple deprecated filter parameters were used:
phase in 0.13 is "zero-double" but will change to "zero" in 0.14
fir_window in 0.13 is "hann" but will change to "hamming" in 0.14
upper transition bandwidth in 0.13 is 0.5 Hz but will change to "auto" in 0.14
The default filter length in 0.13 is "10s" but will change to "auto" in 0.14
```

jupyter ipynb_report Last Checkpoint: 12/08/2016 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Python [default]
Code
CellToolbar
Figure 1
:/DATAS/motorprobe/motor_ica_pipeline/_keys_butorina_anna__131105__CONTROL_RH__AB_control_RH_raw_tsss_-fif/ica/AB_control_RH_raw
CA 001
CA 002
CA 003
CA 004
CA 005
CA 006
CA 007
CA 008
CA 009
CA 010
CA 011
CA 012
CA 013
CA 014
CA 015
CA 016
CA 017
CA 018
CA 019
CA 020
40 42 44 46 48 50
Help
0 100 200 300 400
Time (s)

▸ Link to the notebook

collect_neuropype



```
/media/dmalt/SSD500/aut_gamma/aut_gamma_pipeline
❯ collect_neuropype --help                                          dmalt@LKJ006-psy 14:25:01
Usage: collect_neuropype [OPTIONS] COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...

  Gather neuropype output in pandas dataframe

Options:
  -w, --workflow-path PATH
  -r, --subj-id-regexp TEXT
  -s, --savename PATH
  --help                      Show this message and exit.

Commands:
  conn  Add connectivity matrices
  mse   Add multisale entropy
  pwr   Add powers on trials
```

- Collect output from the pipeline
- Set labels for machine learning
- Save everything in pandas dataframe (big table with data)

Installation

```
$ git clone https://github.com/dmalt/get_some_rest.git
$ cd get_some_rest
$ pip install –editable .
```