MNE-Python: Automatic processing of neurophysiology datasets organized using the Brain Imaging Data Structure (BIDS)

About me

My name is Stefan Appelhoff (stefanappelhoff.com) and I am currently pursuing my PhD at the Max Planck Institute for Human Development. I am enrolled at the Freie Universität Berlin (CEST timezone = UTC+2), and my studies are focused on the cognitive neuroscience of human information search.

For my research, I employ EEG methodology which lead me first to using the MNE-Python analysis software package, and then to contributing to it via GitHub (@sappelhoff).

For more information on my background, activities, and more, please see my CV. Feel free to get in touch via Email as well (stefan.appelhoff@mailbox.org)!

Code contribution

- Exemplary PR to MNE-Python
  - https://github.com/mne-tools/mne-python/pull/5633
- Full list of my PRs to MNE-Python
  - See here

Project information

Sub-org name

MNE-Python

- Website: https://martinos.org/mne/stable/index.html
- GitHub: https://github.com/mne-tools/mne-python

Project Abstract

In this project, we will bring together MNE-Python's standard neurophysiological data processing functions with the Brain Imaging Data Structure (BIDS) to create a fully automated processing pipeline. As an input, the pipeline will require only a minimal configuration file and a BIDS dataset. As an output, the pipeline will provide a path to saved derivative files as well as a detailed report on all processing steps.

Detailed description

Analyzing neurophysiological data involves setting up processing pipelines with common steps such as digital filtering or artifact detection. MNE-Python is a software package that provides a large range of functions to achieve all steps in such analysis pipelines from pre-processing, over analysis, to visualization of the results. In the majority of cases, the processing steps in a pipeline are equal and with only slight modification in parameters.

Despite this situation, fully automated processing pipelines are still rare in the domain of neurophysiological data analysis. Part of the problem is that for a long time, there was no standard on how to organize and document neurophysiological data and thus, researchers have to spend valuable time to "hunt" for metadata as input into the pipeline and fit trivial analysis scripts to the specific organization of raw data in their research lab. Furthermore, the organization of data is typically different from lab to lab research and it can happen that even within a single lab (or a single person over time), different styles of organization are used, exacerbating the problem.
In 2016, the Brain Imaging Data Structure (BIDS) was created to target this problem. BIDS is an emerging standard in the neuroimaging community that allows researchers to neatly organize their data in predefined ways. Next to prescriptions on the organization of data, BIDS requires an exhaustive documentation of metadata for (i) the overall study, (ii) study participants, and (iii) each data recording. These metadata are saved in easily accessible tab-separated files (.tsv) and/or in data dictionaries in the JSON format. The BIDS specification is a manual that describes all metadata that is REQUIRED, or merely RECOMMENDED or OPTIONAL to be present within a given BIDS dataset. Very recently, BIDS has been extended to incorporate neurophysiological modalities such as MEG, EEG, and iEEG.

In summary, automated processing pipelines for neurophysiological data face a problem of having to deal with arbitrary data structures and uncertainty about available metadata. However, this problem can be bypassed with BIDS datasets, that are structured in a principled way and provide full certainty about which metadata must be available.

This project is targeted at bringing together MNE-Python's standard processing functions and the BIDS data structure into a coherent and fully automated data processing pipeline.

For that, we will take inspiration from already existing pipelines and potentially make use of parts of them:

- ephypype
- mne-study-template
- autoreject
- pyprep

Furthermore we will draw on already existing tools for handing BIDS data such as:

- MNE-BIDS
- pybids

The goal is to provide a simple API that allows researchers to feed in a BIDS dataset and a minimal configuration file. As a result, researchers are provided with derivative files, and a structured report about all processing steps. The processing steps that are targeted for now follow the list in Best Practices in Data Analysis and Sharing in Neuroimaging using MEEG:

1. identification and removal of electrodes with poor signal quality
2. artifact identification and removal
3. detrending

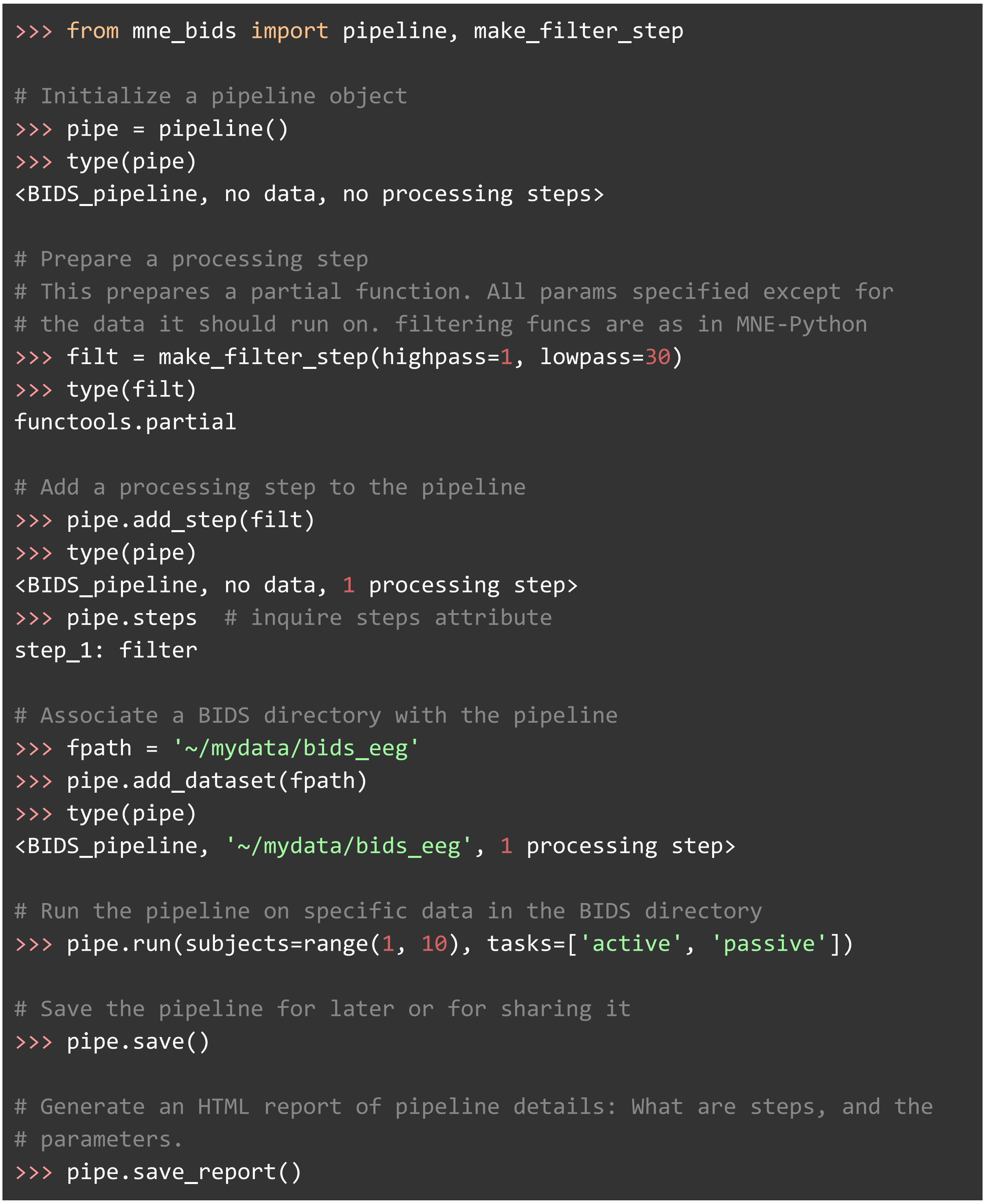
4. digital low- and high-pass filtering
5. data segmentation
6. additional identification/elimination of artifacts
7. baseline correction
8. re-referencing

After these steps, basic analyses will be added:

1. averaging
2. Plotting

The main development of the pipeline will take place in the mne-bids repository, because it's more lightweight compared to the MNE-Python repository and the continuous integration suite can run much faster.

A potential API is given on the next page.



```
>>> from mne_bids import pipeline, make_filter_step

# Initialize a pipeline object
>>> pipe = pipeline()
>>> type(pipe)
<BIDS_pipeline, no data, no processing steps>

# Prepare a processing step
# This prepares a partial function. All params specified except for
# the data it should run on. filtering funcs are as in MNE-Python
>>> filt = make_filter_step(highpass=1, lowpass=30)
>>> type(filt)
functools.partial

# Add a processing step to the pipeline
>>> pipe.add_step(filt)
>>> type(pipe)
<BIDS_pipeline, no data, 1 processing step>
>>> pipe.steps  # inquire steps attribute
step_1: filter

# Associate a BIDS directory with the pipeline
>>> fpath = '~/mydata/bids_eeg'
>>> pipe.add_dataset(fpath)
>>> type(pipe)
<BIDS_pipeline, '~/mydata/bids_eeg', 1 processing step>

# Run the pipeline on specific data in the BIDS directory
>>> pipe.run(subjects=range(1, 10), tasks=['active', 'passive'])

# Save the pipeline for later or for sharing it
>>> pipe.save()

# Generate an HTML report of pipeline details: What are steps, and the
# parameters.
>>> pipe.save_report()
```

Weekly timeline

Community Bonding (May 7-26)

- I will join the MNE-Python Coding Sprint from 22nd to 26th of April 2019
    - This will provide ample opportunity to get to know each other in person, discuss the project, and identify additional and/or alternative tasks
- Learn more about autoreject, ephypype, and the mne-study-template
- Learn more about MNE-Reports for making html summaries of EEG data analysis

Week 1 (May 27-31)

- Topic: “identification and removal of electrodes with poor signal quality”
- First using autoreject on single subject data
- Then on a whole BIDS directory … save autoreject reports
- Compare autoreject “bad channels” with bad channels as marked in BIDS structure
- **Deliverable**: Process BIDS data in auto reject and provide the “diff” of bad channels as marked by autoreject and as documented in BIDS metadata (“channels.tsv” file)

Week 2 (June 3)

- Topic: “artifact identification and removal”
- Run ICA on BIDS dataset and save ICA objects
- Based on available data, try to find artifacts
    - E.g., if “EOG” channels in BIDS data (see channels.tsv), use that channel to find EOG artifacts (see MNE-example)
- **Deliverable**: Function that takes a BIDS dataset and performs ICA for all subjects … data is saved in a /derivatives directory

Week 3 (June 10)

- I will not be able to write code in this week, but I am available via standard communication channels (see other commitments)
- **Deliverable**: n/a

Week 4 (June 17)

- First two days unavailable (see other commitments)
- Topic “digital low- and high-pass filtering”
- **Deliverable**: Pass a BIDS directory and a configuration file: Automatically filter all subject data and provide a comprehensive report on filtering details according to this paper: Widmann et al. 2015

Week 5 (June 24)

- Topic: “remaining preprocessing”

- Re-referencing, additional identification/elimination of artifacts, line noise removal
- **Deliverable**: functionality for above named processing steps

Week 6 (July 1)

- Topic: “segmentation, baseline correction, and averaging”
- **Deliverable**: function can take config file to perform segmentation, baseline correction, and averaging

Week 7 (July 8)

- Topic: “plotting 1”
- **Deliverable**: initial functionality of plots after pipeline run

Week 8 (July 15)

- Topic: “plotting 2”
- **Deliverable**: full functionality of plots after pipeline run

Week 9 (July 22)

- Buffer time if one week takes longer than expected
- **Deliverable**: n/a

Week 10 (July 29)

- Buffer time if one week takes longer than expected
- **Deliverable**: n/a

Week 11 (August 5)

- Topic: “find non-tested features and add tests”
- … also improve the testing
  - Combine similar tests
  - Extend shallow tests
  - Speed up unnecessarily long tests
- **Deliverable**: Comprehensive test suite

Week 12 (August 12)

- Topic: “document everything”
- Also create some examples, and potentially an interactive example using Binder
- **Deliverable**: Full documentation of everything plus at least one use example

Final week (August 19)

- Topic: “submitting”
- **Deliverable**: The full project

Other commitments

I am the main organizer of the yearly Summer Institute on Bounded Rationality for 2019, which will take place from the 11th until the 19th of June. During this time (7 days), I will not have time to code and pursue my project work. I will however be reachable as usual via Email and other communication channels.

My PhD contract at the Max Planck Institute will be paused for the three months of GSoC, so there are no other commitments for me during this time.