3-6 February 2020 - Athens, Greece

Human Brain Project
SUMMIT & OPEN DAY

Human Brain Project

Developing pipelines for multi- scale/species/method analysis

adaptable, reusable, UseCase-driven

05.02.2020 HBP Summit, *Athens*

Robin Gutzen, *Institute of Neuroscience and Medicine (INM-6), FZJ*

JÜLICH Forschungszentrum

Co-funded by the European Union

Human Brain Project

Where we are - Where we want to be

Towards large research collaborations and long-lasting projects

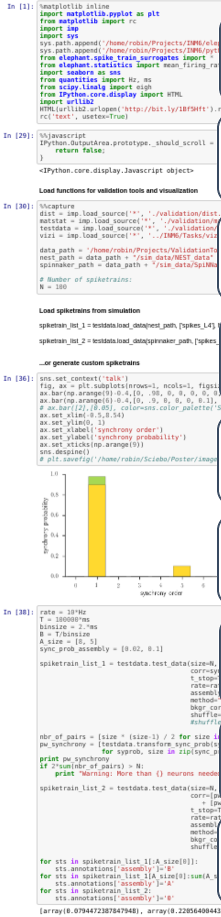
- Data sharing becomes easier and more common
- Open Access becomes more prominent
- "Reproducibility Crisis"

- Easy publishing of analysis workflows
- Easy replication of published results
- Easy reuse of published data/results/analysis

How can we adapt our computational research practices?

- There is no one-size-fit-all solution
- A lot of little, incremental improvements
- Many solutions and tools already exist
- There is progress in bringing tools together
- There is progress in co-developing new features and bridges

Human Brain Project



```
In [1]:  %matplotlib inline
         import matplotlib.pyplot as plt
         from matplotlib import rc
         import imp
         import sys
         sys.path.append('/home/robin/Projects/INM6/ele
         sys.path.append('/home/robin/Projects/INM6/pyt
         from elephant.spike_train_surrogates import *
         from elephant.statistics import mean_firing_ra
         import seaborn as sns
         from quantities import Hz, ms
         from scipy.linalg import eigh
         from IPython.core.display import HTML
         import urllib2
         HTML(urllib2.urlopen('http://bit.ly/1Bf5Hft').read())
         rc('text', usetex=True)

In [29]: %%javascript
         IPython.OutputArea.prototype._should_scroll =
             return false;
         }

         <IPython.core.display.Javascript object>
```

Load functions for validation tools and visualization

```
In [30]: %%capture
         dist = imp.load_source('*', './validation/dist.p
         matstat = imp.load_source('*', './validation/m
         testdata = imp.load_source('*', './validation/
         vizi = imp.load_source('*', '../INM6/Tasks/viz

         data_path = '/home/robin/Projects/ValidationTo
         nest_path = data_path + "/sim_data/NEST_data"
         spinnaker_path = data_path + "/sim_data/SpiNNa

         # Number of spiketrains:
         N = 100
```

Load spiketrains from simulation

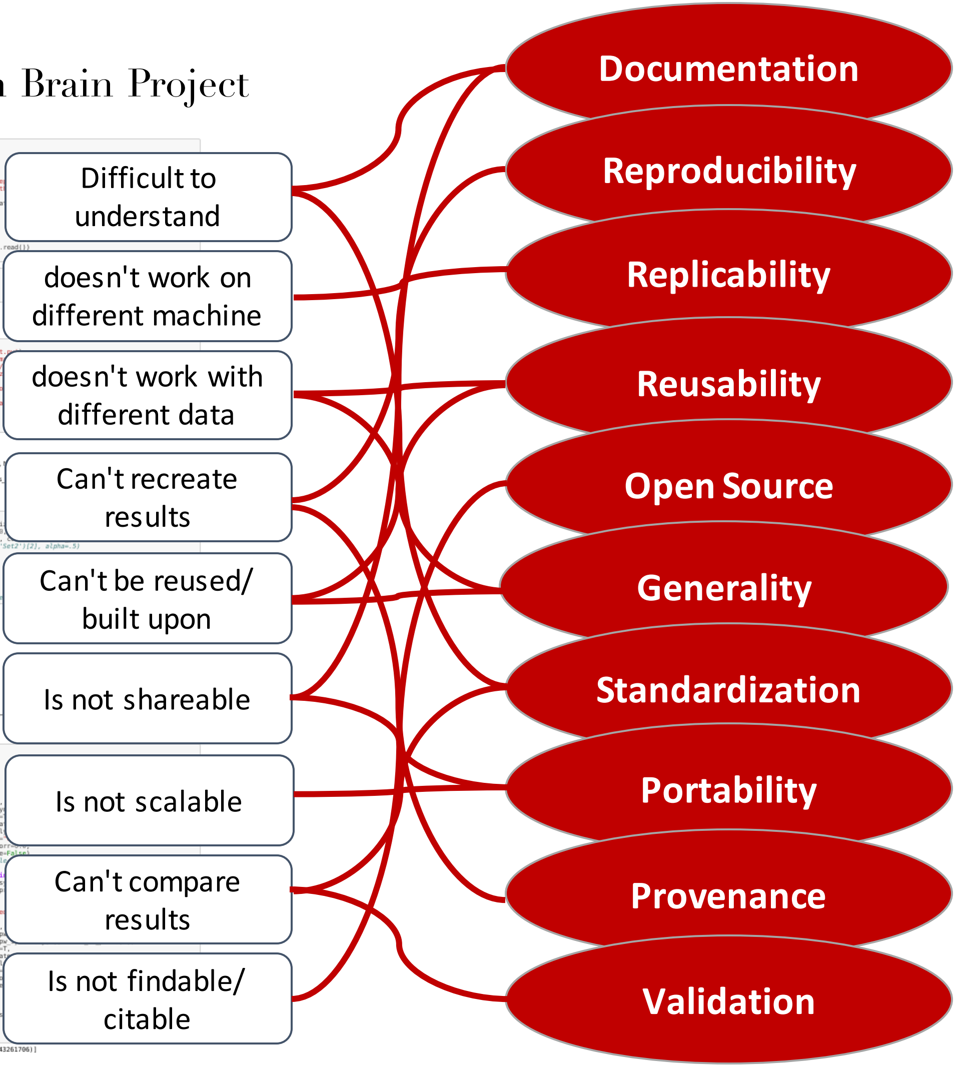
spiketrain_list_1 = testdata.load_data(nest_path, ['spikes_L4'],

spiketrain_list_2 = testdata.load_data(spinnaker_path, ['spikes_

...or generate custom spiketrains

```
In [36]: sns.set_context('talk')
         fig, ax = plt.subplots(nrows=1, ncols=1, figsi
         ax.bar(np.arange(9)-0.4,[0, .98, 0, 0, 0, 0, 0,
         ax.bar(np.arange(6)-0.4,[0, .9, 0, 0, 0, 0.1], c
         # ax.bar([2],[0.05], color=sns.color_palette('Set2')[2], a
         ax.set_xlim(-0.5,8.54)
         ax.set_ylim(0, 1)
         ax.set_xlabel('synchrony order')
         ax.set_ylabel('synchrony probability')
         ax.set_xticks(np.arange(9))
         sns.despine()
         # plt.savefig('/home/robin/Sciebo/Poster/image
```

```
In [38]: rate = 10*Hz
         T = 100000*ms
         binsize = 2.*ms
         B = T/binsize
         A_size = [8, 5]
         sync_prob_assembly = [0.02, 0.1]

         spiketrain_list_1 = testdata.test_data(size=N,
                                                corr=sy
                                                t_stop=
                                                rate=ra
                                                assembl
                                                method=
                                                bkgr_corr=
                                                shuffle=False)
                                                #shuffle

         nbr_of_pairs = [size * (size-1) / 2 for size i
         pw_synchrony = [testdata.transform_sync_prob(sy
                         for syprob, size in zip(sync_p
         print pw_synchrony
         if 2*sum(nbr_of_pairs) > N:
             print "Warning: More than {} neurons needed

         spiketrain_list_2 = testdata.test_data(size=N,
                                                corr=[pw
                                                   + [pw
                                                t_stop=T,
                                                rate=rat
                                                assembl
                                                method=
                                                bkgr_co
                                                shuffle

         for sts in spiketrain_list_1[:A_size[0]]:
             sts.annotations['assembly']='B'
         for sts in spiketrain_list_1[A_size[0]:sum(A_s
             sts.annotations['assembly']='A'
         for sts in spiketrain_list_2:
             sts.annotations['assembly']='B'

[array(0.0794472387847948), array(0.2285640844326170б)]
```



SPHINX Python Documentation Generator

Read the Docs

CONDA

Sumatra

PyPi

zenodo

SciPy.org

PEP

elephant ELECTROPHYSIOLOGY ANALYSIS TOOLKIT

odML

NumPy

docker

neo

Fairgraph

binder

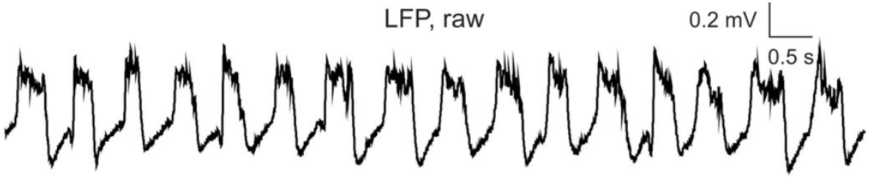
SciUnit

NetworkUnit

Slow Wave Analysis (SP3 UseCase002)

Slow Waves

- periodic transition between Up and Down state (0-5Hz)
- in sleep and anasthesia
- across measurements and species
- relevance for memory and consciousness research

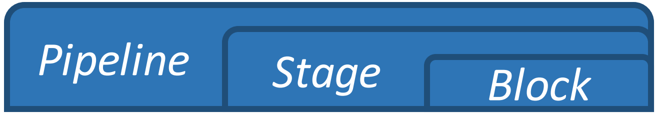


Szabó et. al. 2017

Analysis Pipeline

- identifying a generalized structure of analysis steps

  -> sequential *Stages*, and modular *Blocks*
- implemention with a workflow management sytem
- precisely defining the interfaces and (meta-)data requirements
- integrating existing analysis approaches & algorithms



Human Brain Project

Flowchart

Pipeline

Stage

Block

Calcium Imaging

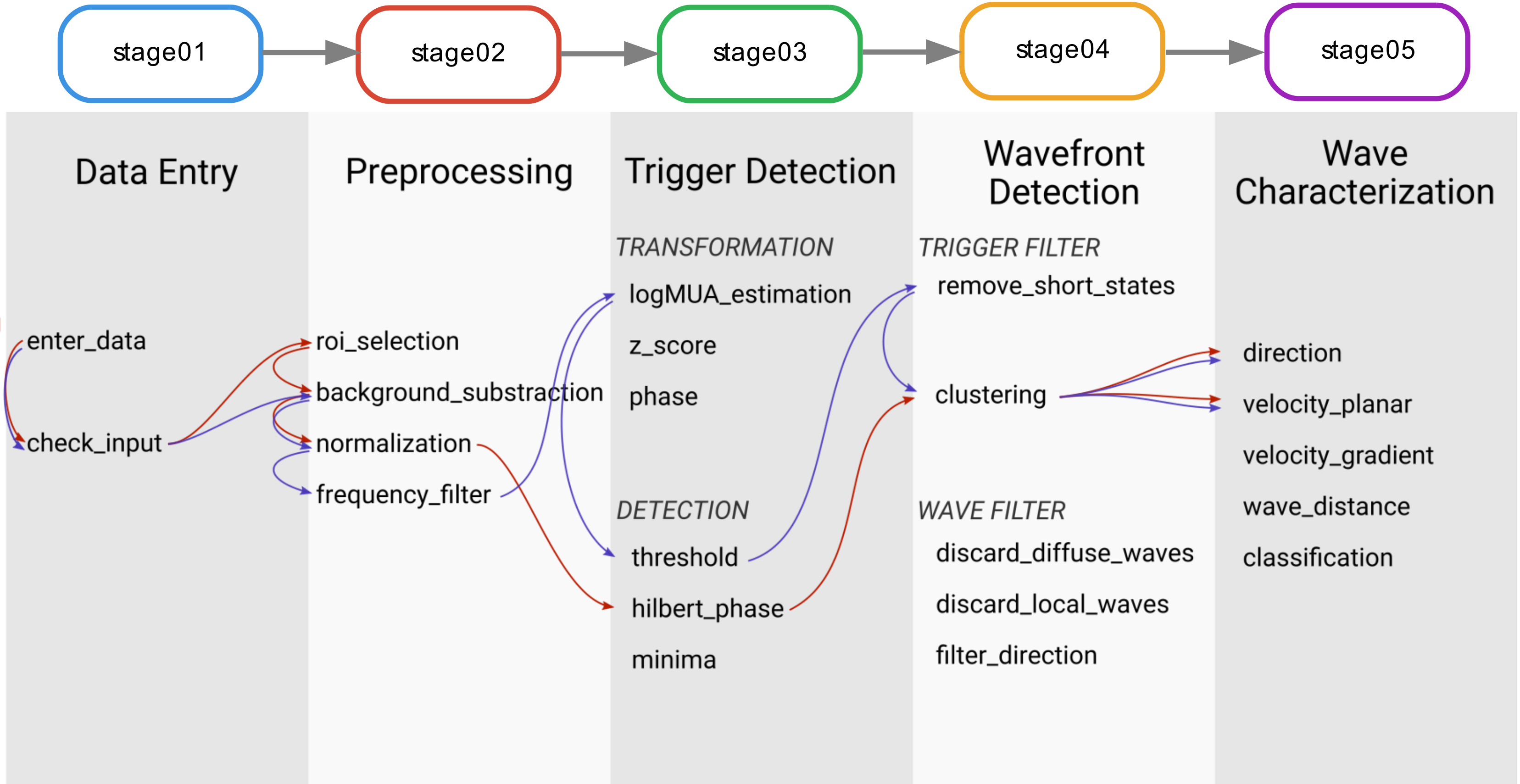
ECoG

EEG

LFP

Spikes

Simulation



stage01 → stage02 → stage03 → stage04 → stage05

Data Entry

enter_data

check_input

Preprocessing

roi_selection

background_substraction

normalization

frequency_filter

Trigger Detection

TRANSFORMATION

logMUA_estimation

z_score

phase

DETECTION

threshold

hilbert_phase

minima

Wavefront Detection

TRIGGER FILTER

remove_short_states

clustering

WAVE FILTER

discard_diffuse_waves

discard_local_waves

filter_direction

Wave Characterization

direction

velocity_planar

velocity_gradient

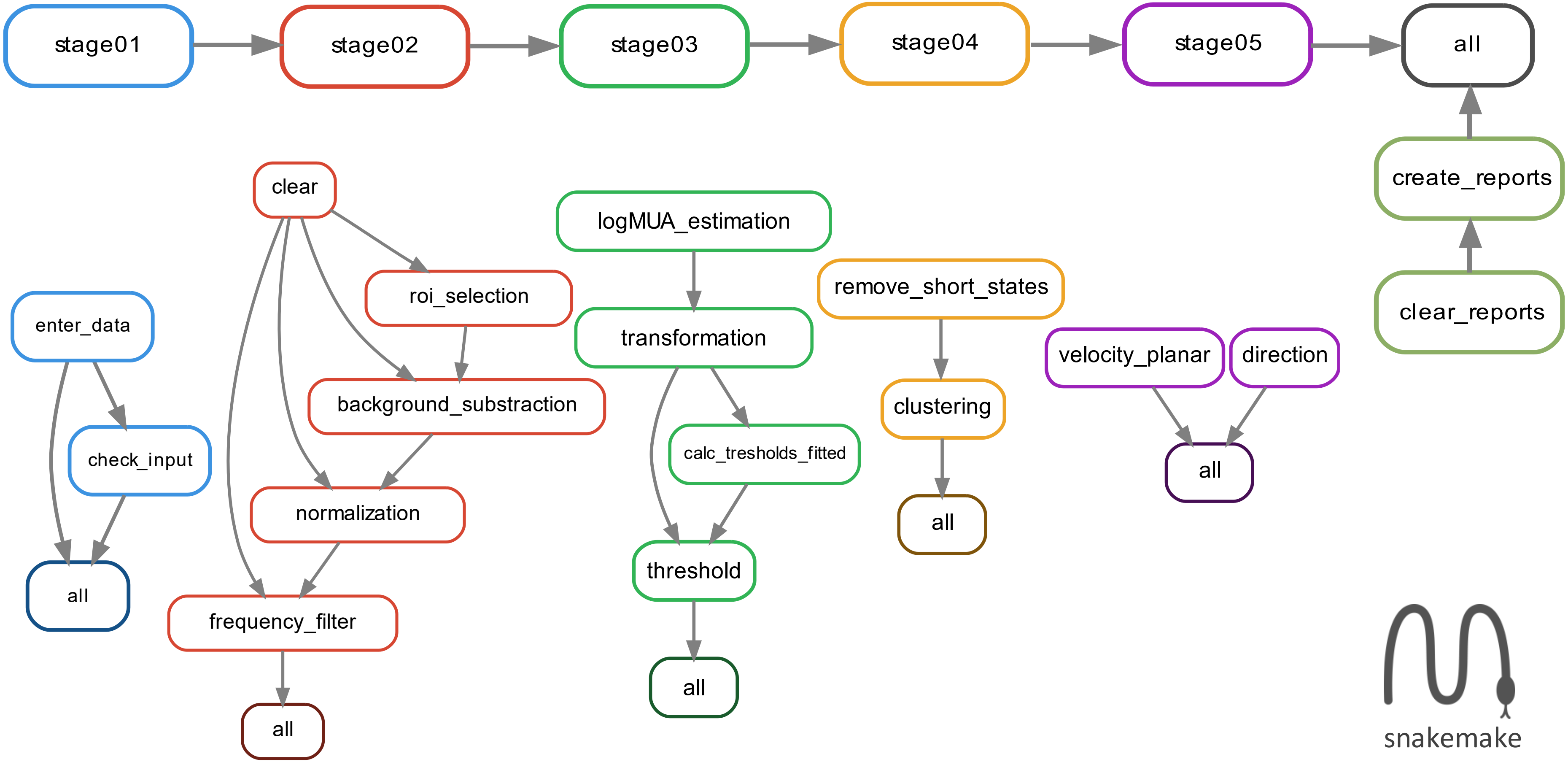
wave_distance

classification

Data curtesy of: F.S. Pavone Lab, LENS, Florence, Italy / M.V. Sanchez-Vives Lab, IDIBABS, Barcelona, Spain

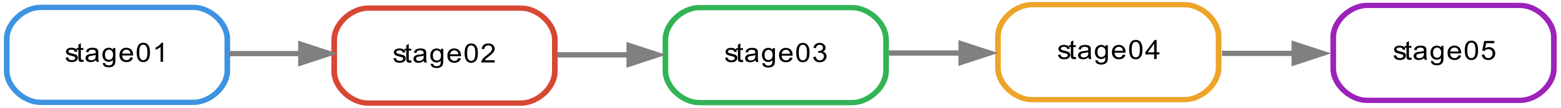
Human Brain Project

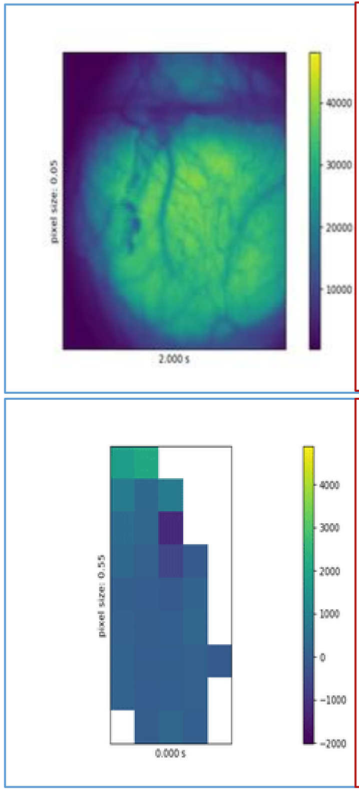
Dependency Trees

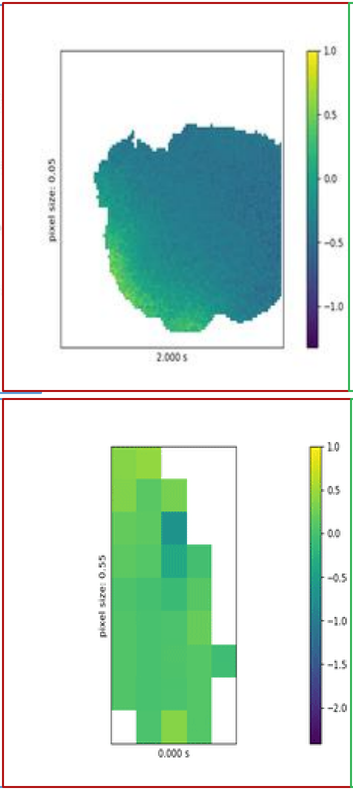


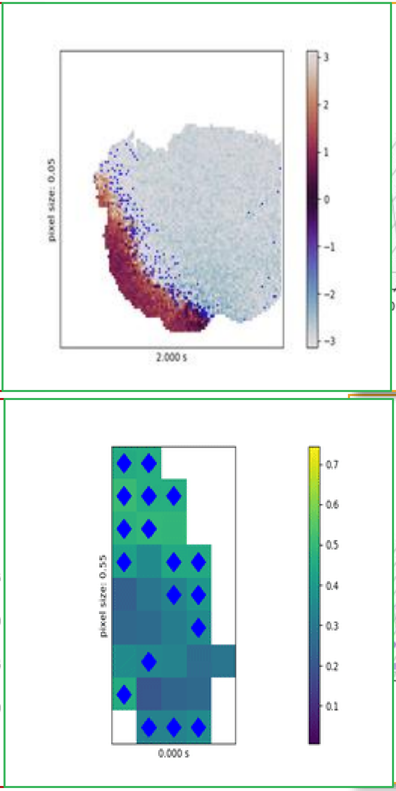
snakemake

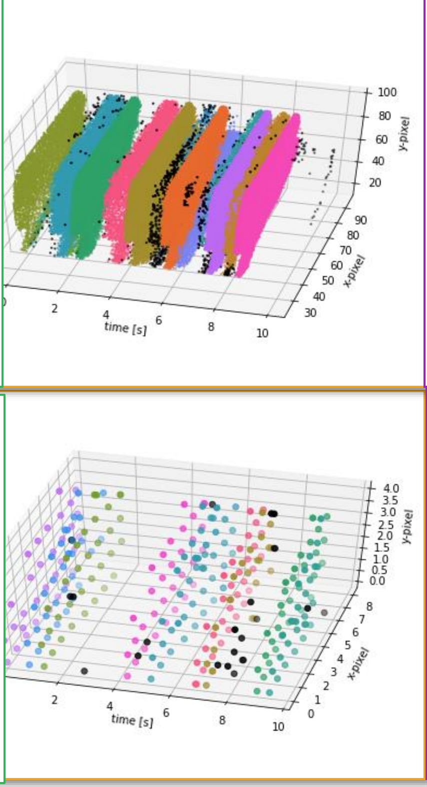
Stage Outputs

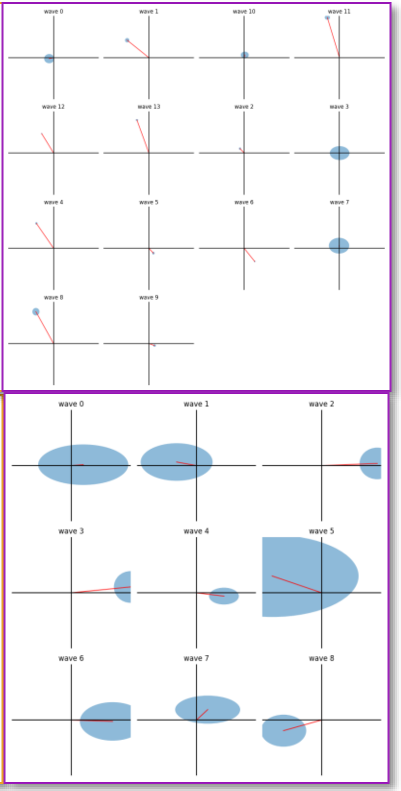






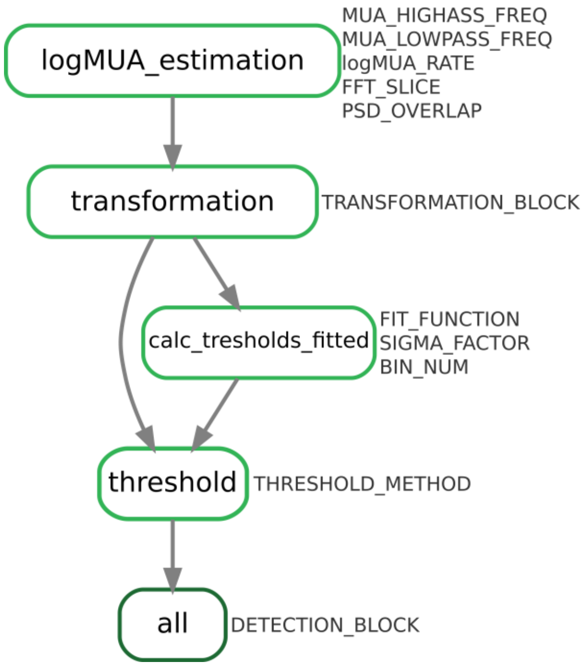


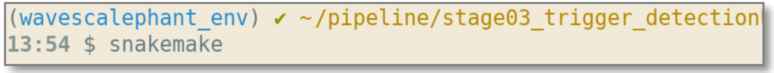




Folder Hierachy

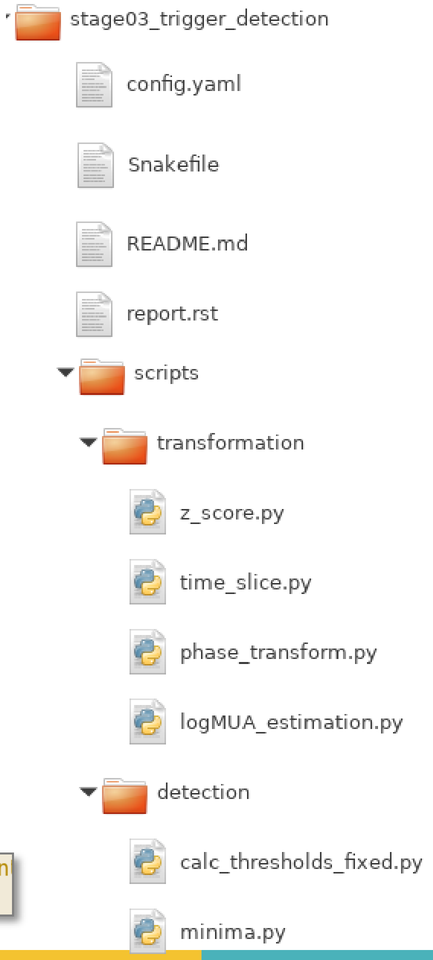
Pipeline | **Stage** | Block





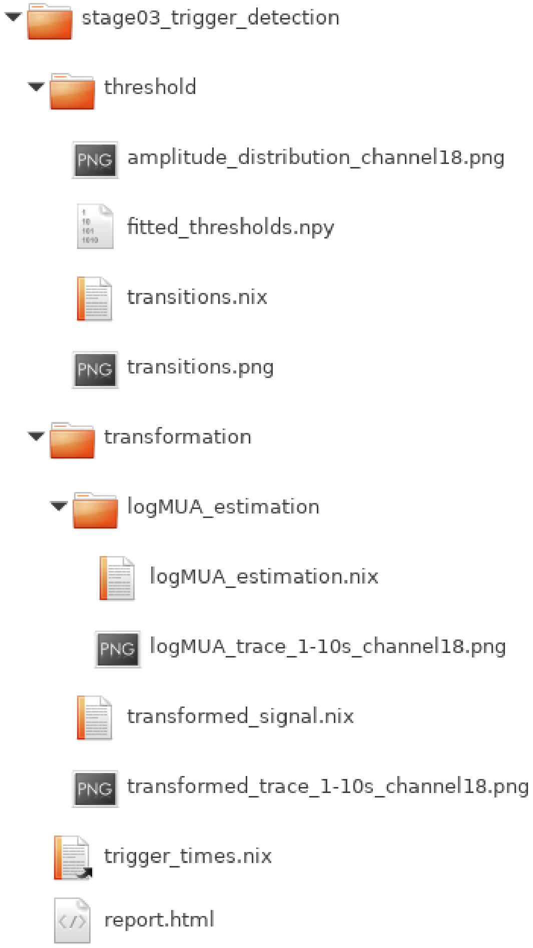
```
(wavescalephant_env) ✔ ~/pipeline/stage03_trigger_detection
13:54 $ snakemake
```

Source Files



- stage03_trigger_detection
  - config.yaml
  - Snakefile
  - README.md
  - report.rst
  - scripts
    - transformation
      - z_score.py
      - time_slice.py
      - phase_transform.py
      - logMUA_estimation.py
    - detection
      - calc_thresholds_fixed.py
      - minima.py

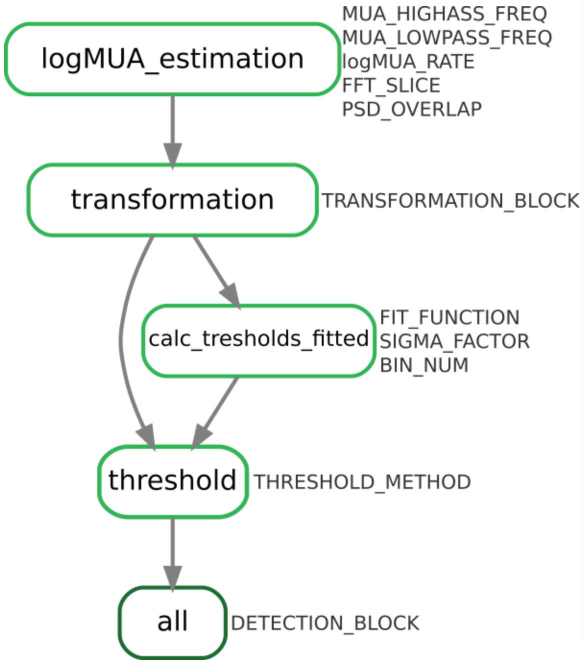
Output Files

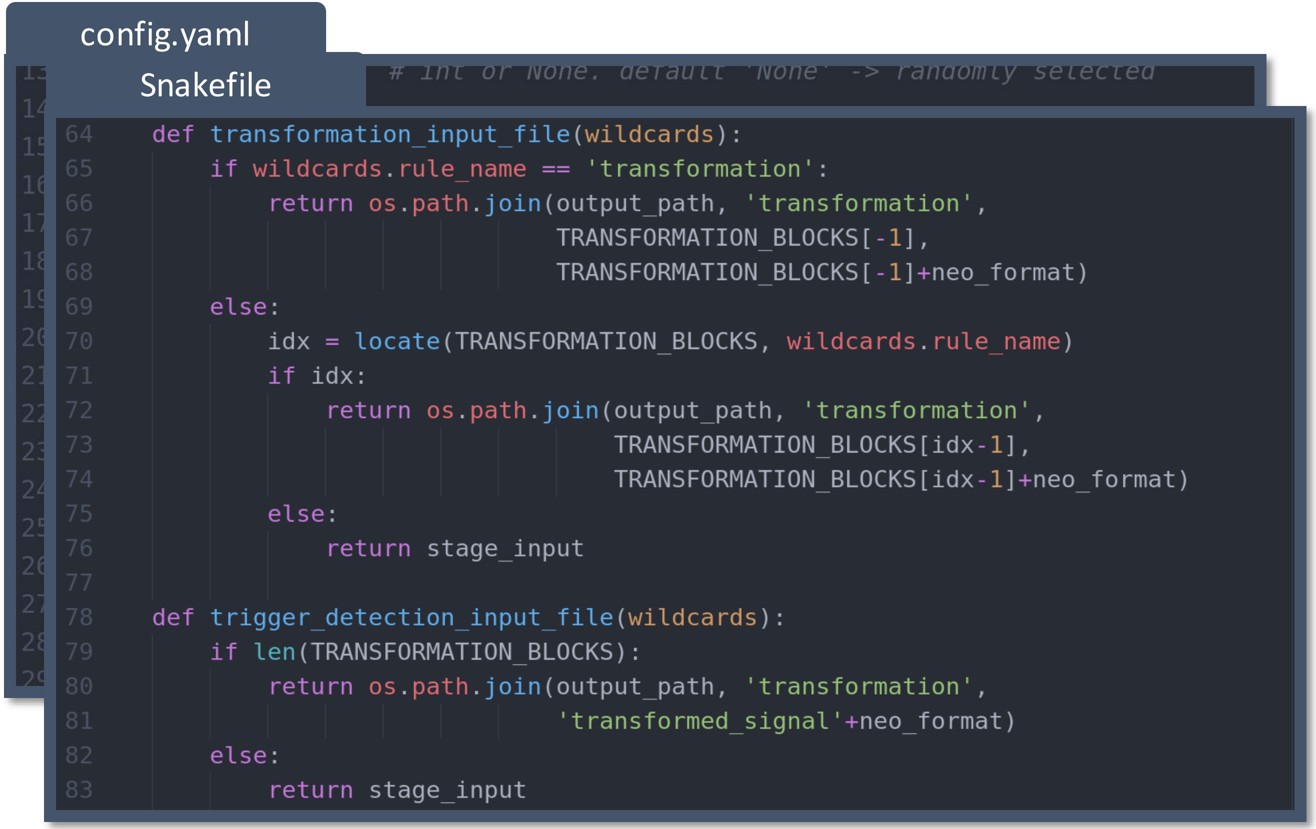


- stage03_trigger_detection
  - threshold
    - amplitude_distribution_channel18.png
    - fitted_thresholds.npy
    - transitions.nix
    - transitions.png
  - transformation
    - logMUA_estimation
      - logMUA_estimation.nix
      - logMUA_trace_1-10s_channel18.png
    - transformed_signal.nix
    - transformed_trace_1-10s_channel18.png
  - trigger_times.nix
  - report.html

Config & Snakefile

How to flexibly adapt the trigger detection method?

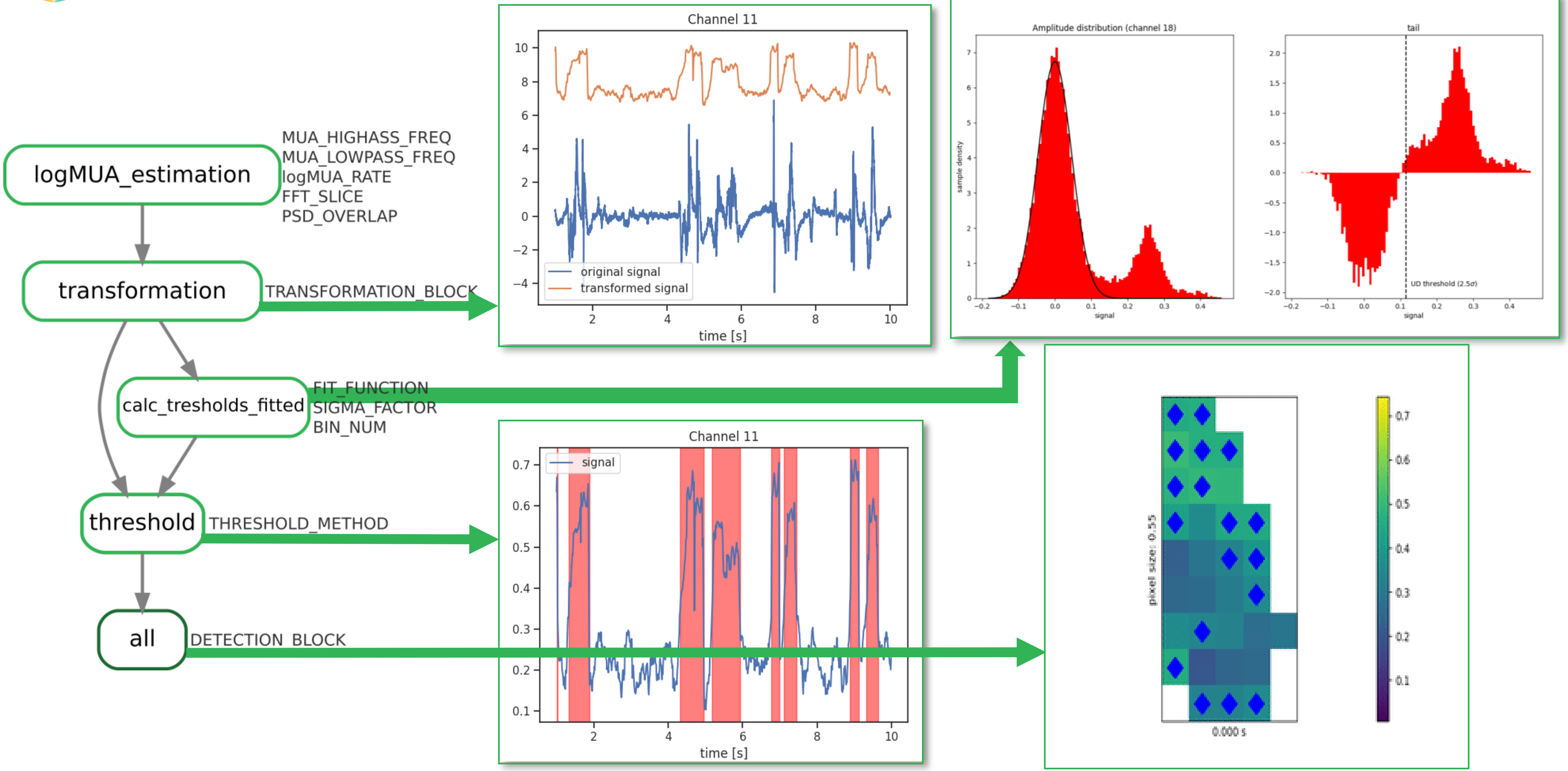




```
# int or None. default 'None' -> randomly selected
```

```
def transformation_input_file(wildcards):
    if wildcards.rule_name == 'transformation':
        return os.path.join(output_path, 'transformation',
                            TRANSFORMATION_BLOCKS[-1],
                            TRANSFORMATION_BLOCKS[-1]+neo_format)
    else:
        idx = locate(TRANSFORMATION_BLOCKS, wildcards.rule_name)
        if idx:
            return os.path.join(output_path, 'transformation',
                                TRANSFORMATION_BLOCKS[idx-1],
                                TRANSFORMATION_BLOCKS[idx-1]+neo_format)
        else:
            return stage_input

def trigger_detection_input_file(wildcards):
    if len(TRANSFORMATION_BLOCKS):
        return os.path.join(output_path, 'transformation',
                            'transformed_signal'+neo_format)
    else:
        return stage_input
```

Block Outputs



logMUA_estimation

MUA_HIGHASS_FREQ
MUA_LOWPASS_FREQ
logMUA_RATE
FFT_SLICE
PSD_OVERLAP

transformation

TRANSFORMATION_BLOCK

calc_tresholds_fitted

FIT_FUNCTION
SIGMA_FACTOR
BIN_NUM

threshold

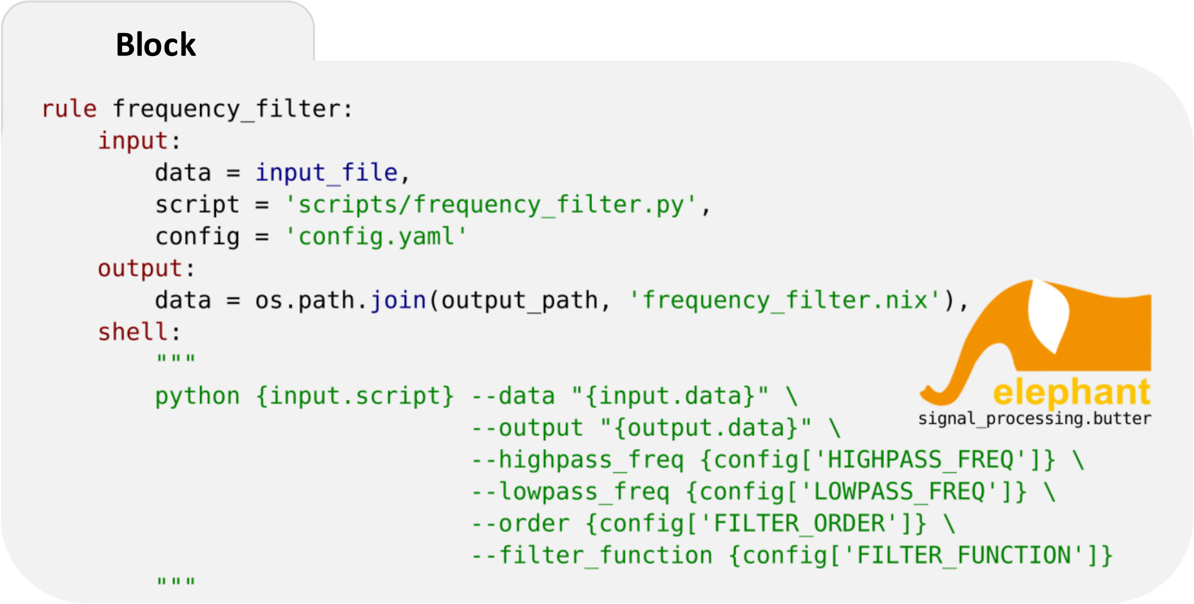
THRESHOLD_METHOD

all

DETECTION_BLOCK

Block Structure

Block = Instructions to create output from input (*snakemake rule*)



Block

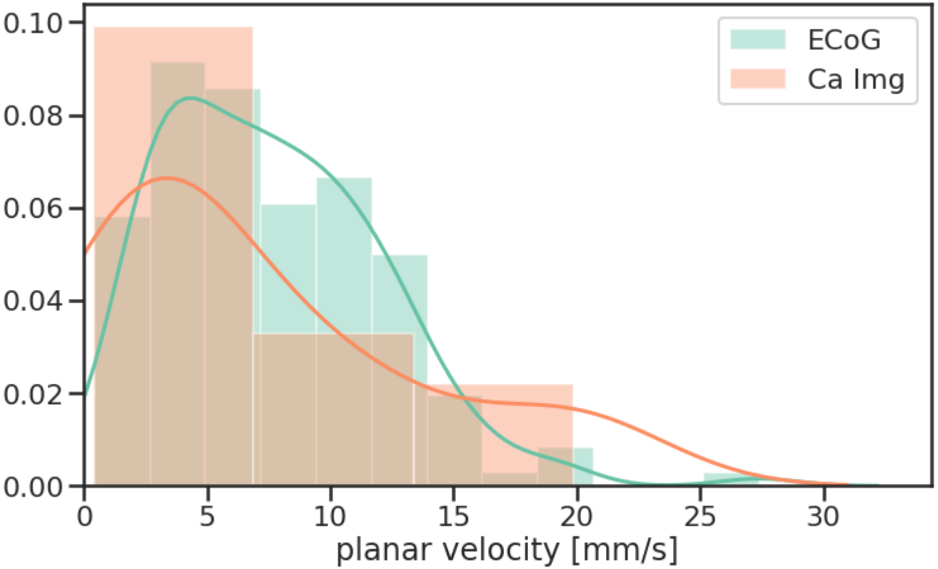
```
rule frequency_filter:
    input:
        data = input_file,
        script = 'scripts/frequency_filter.py',
        config = 'config.yaml'
    output:
        data = os.path.join(output_path, 'frequency_filter.nix'),
    shell:
        """
        python {input.script} --data "{input.data}" \
                              --output "{output.data}" \
                              --highpass_freq {config['HIGHPASS_FREQ']} \
                              --lowpass_freq {config['LOWPASS_FREQ']} \
                              --order {config['FILTER_ORDER']} \
                              --filter_function {config['FILTER_FUNCTION']}
        """
```

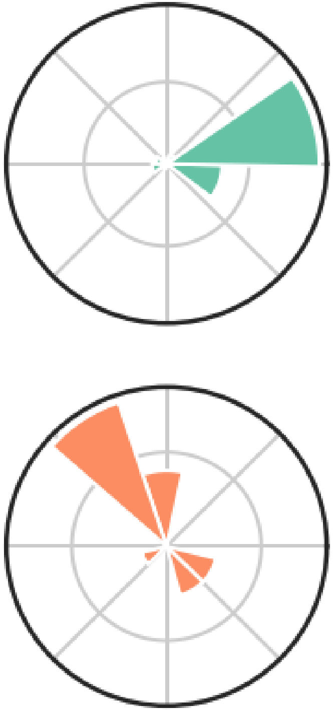
Snakemake Advantages

- Can use any shell executables
- Handles portability
- Determines execution order
- Captures execution details

The Payoff

Direct comparability between different data types

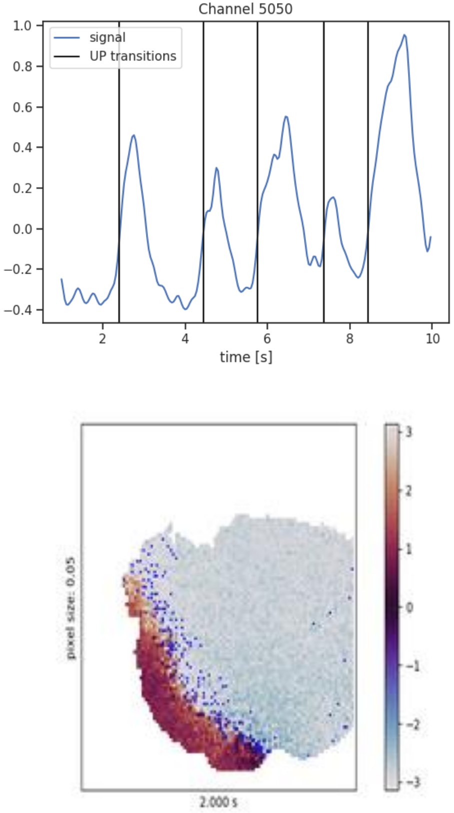




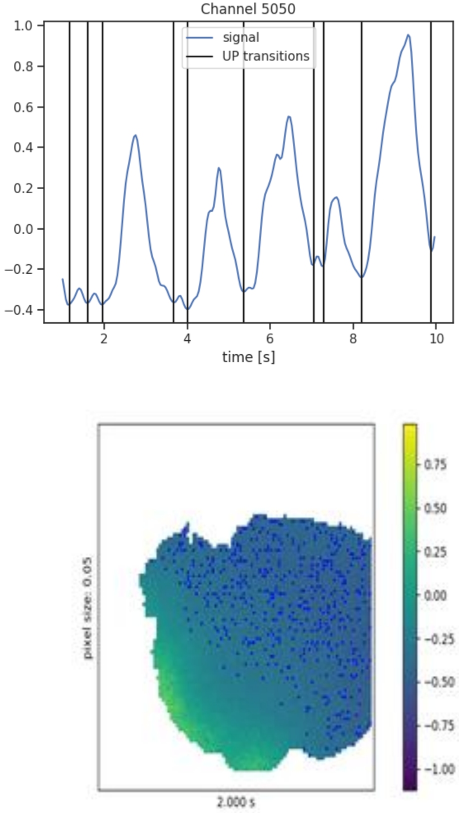
The Payoff

Benchmarking of analysis algorithms

via Hilbert phase



via minima



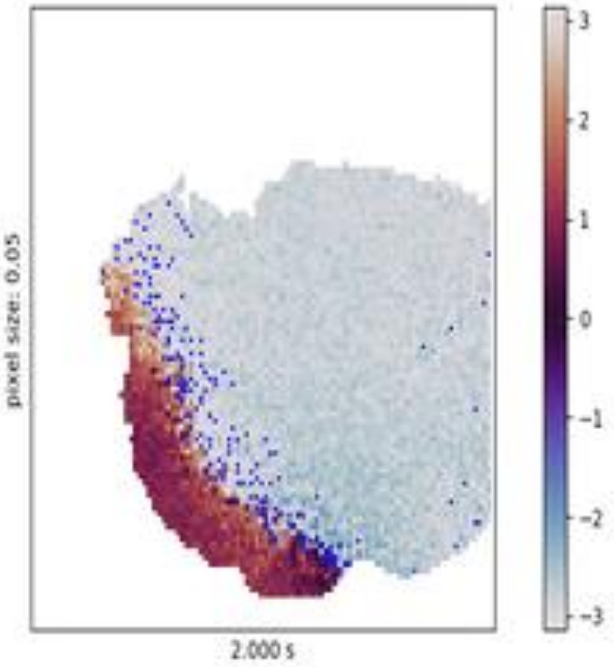
Human Brain Project

The Payoff

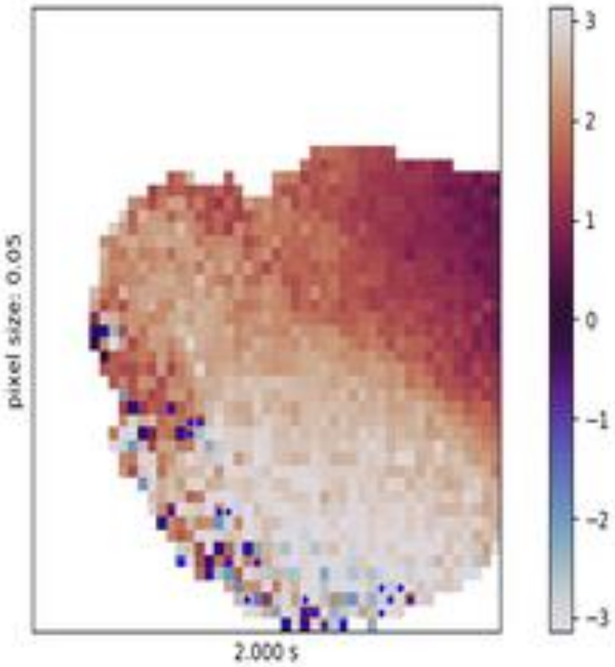
Basis for meaningful validation tests

NetworkUnit SciUnit

recorded activity

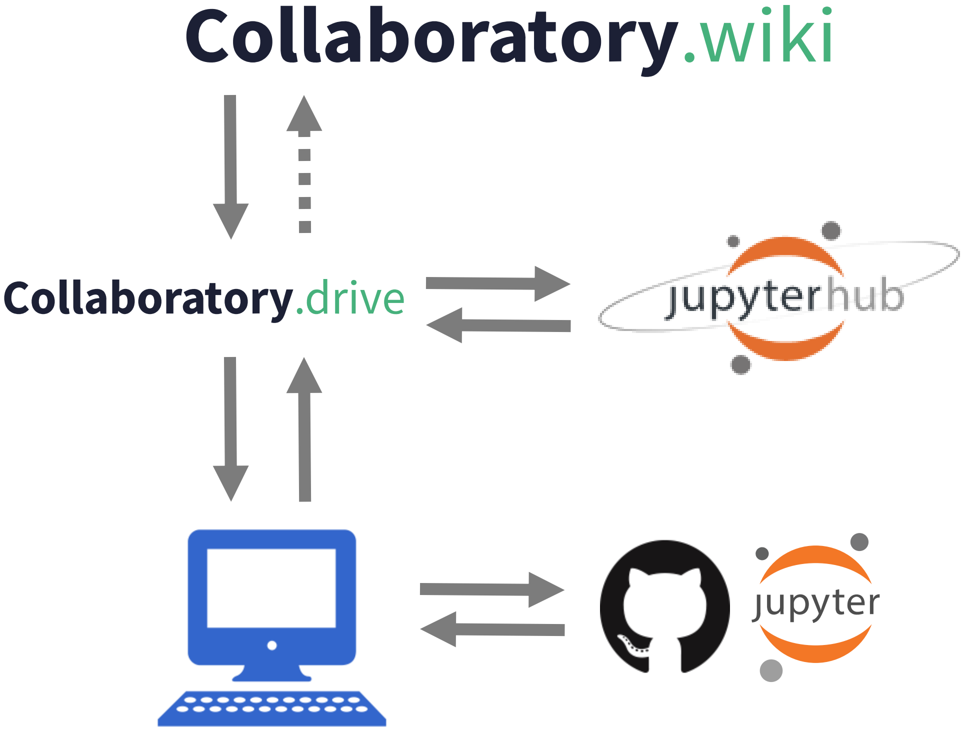


simulated activity



Model by Cristiano Capone

Collab Integration



Wishlist

- GitHub Integration / Version Control
- (Python) Environment Management
- Easy HPC Access
- Workflow Engine

Human Brain Project

Thank you!

Giulia De Bonis

Cristiano Capone

Elena Pastorelli

Pier Stanislao Paolucci

Yann Zerlaut

Glynis Mattheis

Andrew Davison

Robin Gutzen

Michael Denker

JÜLICH Forschungszentrum

INFN Istituto Nazionale di Fisica Nucleare

APE

UNIC

IDIBAPS

Lens European Laboratory for Non-Linear Spectroscopy

UNIVERSITÀ DEGLI STUDI FIRENZE

THANK YOU!

www.humanbrainproject.eu

@humanbrainproj

@humanbrainproj

HumanBrainProject

Co-funded by
the European Union