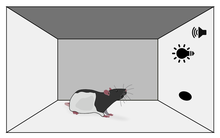
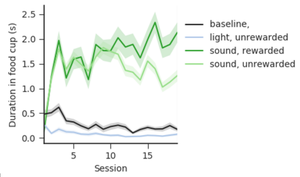
Moving Neuroscience Forward with Python

Emily Irvine and Matthijs van der Meer
Dartmouth College, Hanover NH

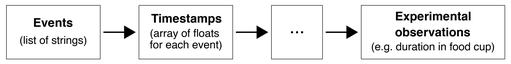
Recording animal behavior



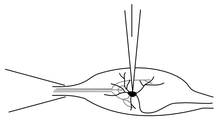
Negative occasion setting
We give the same cue different meanings in different contexts to see how animals react to the cue in those contexts.



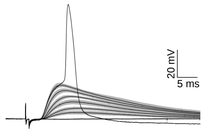
Rats respond in the rewarded context
This plot aggregates data across 8 subjects.
The filled regions are 95% confidence intervals.



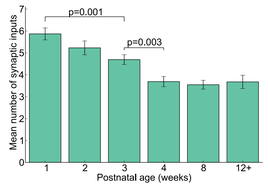
Recording electrical activity in a single cell



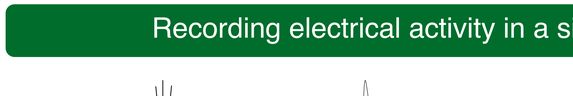
Intracellular recordings
We stimulate the superior cervical ganglion's preganglionic nerve and record from individual neurons in the superior cervical ganglion with a sharp electrode.



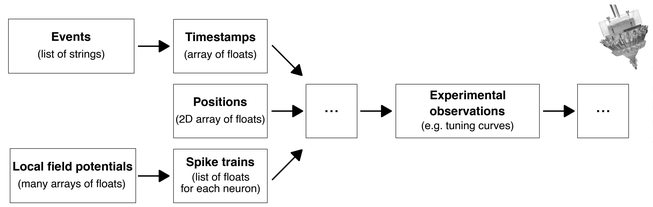
A neuron with many synaptic inputs
Membrane potential of a neuron when stimulated with gradually increasing current. We can estimate the number of synaptic inputs that each neuron receives using k-means clustering and statistical tools.



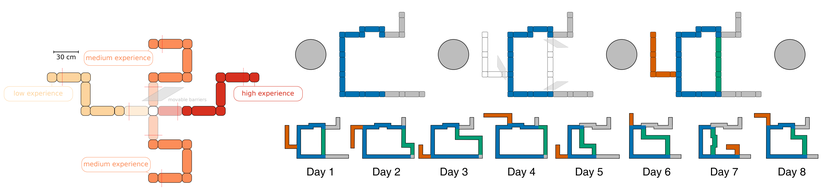
Number of synaptic inputs
Neurons in younger mice have more synaptic inputs than neurons in older mice. This is due to a process known as synaptic refinement.



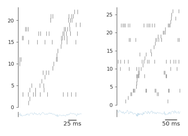
Recording many cells in freely moving animals



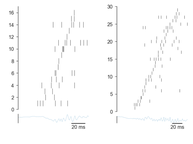
In-vivo recordings
We simultaneously record many neurons in the hippocampus of freely moving rats on complex and changing mazes using 16-tetrode hyperdrives.



Tuning curves
(left) Each heatmap shows the firing rate of a neuron when the rat was at each location in the maze. These neurons tend to fire most at certain locations, and are known as *place cells*.



Spike rasters
(right) Each plot shows the spiking activity of many neurons (one neuron for each row of spikes) ordered based on their preferred firing location in the maze when a rat runs **(top)** on the maze and rests **(bottom)**.



Research shown here was conducted in collaboration with Ellis Cooper (McGill University), Ivy Shen and Leeza Petrov (Dartmouth College).
Contact info: emilymirvine@gmail.com

Neuroscienctific experiments produce widely varying amounts of data. Behavioral experiments record an event every few seconds, while a multicellular recording experiment samples data from hundreds of channels thousands of times per second. Despite this variability, data is almost always received as strings or floats and stored as arrays of numbers, making Python an ideal choice to run and analyze any neuroscientific experiment. Here, we show examples from three separate experiments where we used Python to deal with different amounts and types of data.

We developed NeuroElectroPhysiology Tools (nept) as a minimalist framework for analyzing experiments like those presented here. Our goal is to provide a readable, well tested, and well documented library that does not tie you to a particular framework and file format. Instead, we provide a few classes that help researchers avoid common pitfalls like slicing data but not time and not considering edge cases like the first and last time periods of an event. nept is pip-installable through PyPI and has its documentation on ReadTheDocs and tests running on TravisCI, though it is small and readable enough that users can copy the code directly into their own scripts.

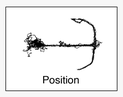
nept is small and readable in part because it relies on several robust and mature tools in the scientific Python ecosystem. These tools have enabled many groups to make contributions to different subfields of neuroscience by releasing their research as open source software packages. Python's readability, well designed testing tools, and packaging and installation infrastructure has advanced the field and improved the quality of neuroscientific resarch results in underappreciated but critical ways. Here, we show a sample of other tools that have been developed to address some of the needs of neuroscientific research.

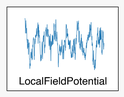
nept : NeuroElectroPhysiology Tools

AnalogSignal

data, time

dimensions
isempty
n_samples
time_slice(t_starts, t_stops)





x, y
distance(position)
linearize(ideal_path, zone)
speed(t_smooth)

Epoch

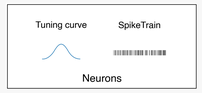
starts, stops

centers
durations
isempty
n_epochs
start
stop
copy()
contains(value, edge)
excludes(epoch)
expand(amount, direction)
intersect(epoch)
join(epoch)
merge(gap)
overlaps(epoch)
shrink(amount, direction)
time_slice(t_starts, t_stops)

SpikeTrain

time, label

n_spikes
time_slice(t_starts, t_stops)



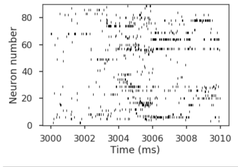
n_neurons
tuning_shape
time_slice(t_starts, t_stops)

www.github.com/vandermeerlab/nept
https://vandermeerlab.github.io/nept

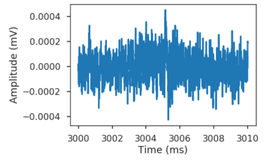
Real world example

```
In [1]: # import some of the necessary packages
        %matplotlib inline
        import os
        import numpy as np
        import matplotlib.pyplot as plt
        import nept

        from loading_data import get_data

In [2]: # import the session-specific info file
        import info.r068d5 as info

In [3]: # Define times of interest
        epoch = nept.Epoch(3000, 3010)

In [4]: # Load the data
        events, position, spikes, lfp, _ = get_data(info)

In [5]: # events is a dictionary of timestamps of key task events
        print("Feeder 1:", events["feeder1"])

        Feeder 1: [ 1431.508077  1506.265639  1561.425108  1618.657514  1684.902733
          1782.222389  1819.075545  1862.49817   1895.375545  1936.104233
          2798.509639  3117.12742   3190.382733  3268.675358  3378.172608
          3456.044795  3558.862764  3648.932483  3716.469702  3785.539014
          3867.346733  5906.801327  5965.726264  6016.990202  6053.943545
          6104.816733  6154.53842   6215.526295  6335.949108  6374.484764
          6418.94892   6468.420295  6514.136233  6570.11692   6621.090452
          6663.802077  6733.912827  6766.940608  6834.848545  6879.683045
          6925.008483  7021.607483  7082.885702  7162.820764  7216.958858
          7276.304108  7332.324764  7371.441139  7448.732483  7541.465577
          7604.826952  7644.524295  7683.100045  7715.537014  7771.157077
          7805.85717   7958.726483  8008.698483  8133.067139]

In [6]: # position contains x, y, and time data of the rat's location
        sliced_position = position.time_slice(epoch.starts, epoch.stops)

        plt.plot(position.x, position.y, "k.", ms=1)
        plt.plot(sliced_position.x, sliced_position.y, "r.", ms=5)
        plt.axis("off")
        plt.show()
```

```
In [7]: # spikes is a list of spiketrains, which are timestamps of when each neuron fires an action potential
        sliced_spikes = [spiketrain.time_slice(epoch.starts, epoch.stops) for spiketrain in spikes]

        location = 1
        for neuron in sliced_spikes:
            if neuron.n_spikes > 0:
                plt.plot(neuron.time, np.ones(neuron.n_spikes)*location, '|', color='k', ms=4, mew=1)
                location += 1
        plt.xlabel('Time (ms)')
        plt.ylabel('Neuron number')
        plt.ylim(0, location+1)
        plt.show()
```




```
In [8]: # lfp contains data and time information of the local field potential
        sliced_lfp = lfp.time_slice(epoch.starts, epoch.stops)

        plt.plot(sliced_lfp.time, sliced_lfp.data)
        plt.xlabel('Time (ms)')
        plt.ylabel('Amplitude (mV)')
        plt.show()
```



Scientific Python ecosystem

NumPy
SciPy
jupyter
ANACONDA
pandas
matplotlib
Seaborn
Shapely
PYTORCH
scikit learn
TensorFlow

Other electrophysiology projects

neo — Neo is a framework for working with electrophysiology data in Python that reads many different file formats and provides a comprehensive object model.
https://neo.readthedocs.io/en/latest/

elephant — Elephant is an electrophysiology analysis toolbox based on the Neo framework.
https://elephant.readthedocs.io/en/latest/

NeuroML — NeuroML provides an object model for describing neuronal morphologies, ion channels, synapses and 3D network structure.
https://neuralensemble.org/libNeuroML/

SpykeViewer — SpykeViewer is a data analysis platform for electrophysiology and neural simulation data that includes a graphical interface and a plugin architecture.
http://spyke-viewer.g-node.org/

Tridesclous — Tridesclous is spike sorting tool using modern Python features, including spike sorting within a Jupyter notebook.
https://tridesclous.readthedocs.io/en/latest/

ephyviewer — Ephyviewer is a simple graphical interface for visualizing electrophysiological signals, spikes, events, videos, and more.
https://ephyviewer.readthedocs.io/en/latest/index.html

nelpy — Nelpy is an open source package for analyzing neuroelectrophysiology data.
https://github.com/nelpy/nelpy

Neuroimaging

Neuroimaging experiments uses techniques like functional magnetic resonance imaging (fMRI) to record more overall brain activity but with lower spatial and temporal precision than electrophysiology. Like electrophysiology, these experiments deal with large multidimensional arrays of numbers and make use of the scientific Python stack, but they are visualized, analyzed, stored, and shared differently, thus requiring a different set of tools.

PyMVPA
MNE
Nipype: Neuroimaging in Python Pipelines and Interfaces
BrainIAK
Nitime: time-series analysis for neuroscience

http://www.pymvpa.org/
https://nipype.readthedocs.io/en/latest/
http://nipy.org/nitime/
http://martinos.org/mne/stable/index.html
https://brainiak.org/
https://mindboggle.info/

Neural simulators

Neural simulators aim to reproduce the activity of neurons in order to model brain systems and make testable predictions of how those systems are organized and how they respond to behavioral and neurophysiological interventions.

Nengo
BRIAN
NEURON
nest
BindsNET
Topographica

https://www.nengo.ai/
http://briansimulator.org/
https://neuron.yale.edu/neuron/
https://www.nest-simulator.org/
https://neuralensemble.org/PyNN/
https://binds.cs.umass.edu/bindsnet.html
http://ioam.github.io/topographica/