The Neuroscience Lab

A Tour Through the Eyes
of a Pythonista

Nicholas A. Del Grosso
Javier Martinez Alcantara

Javier and Nick: Labmates, Neuroscientists, Pythonistas.

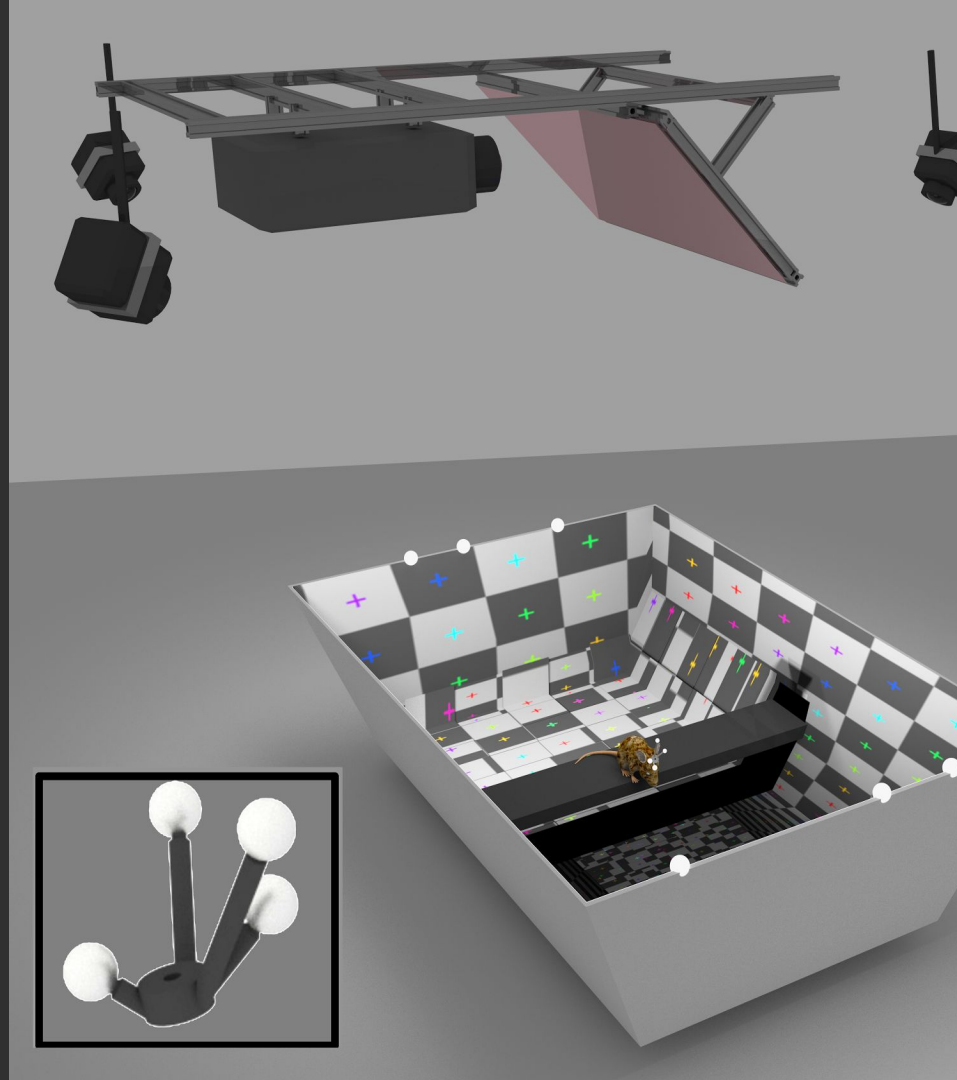
Javier Martinez Alcantara
@jmaralc

Nicholas Del Grosso
@neuroneuro15

Python is Everywhere in Our Lab

- Presenting Visual Stimuli
- Virtual Reality
- Animal Training
- Networking Together Our Devices
- Refine Control of Hardware
- Motion Tracking
- Graphic User Interfaces (GUIs)
- Data Analysis

Collecting Data

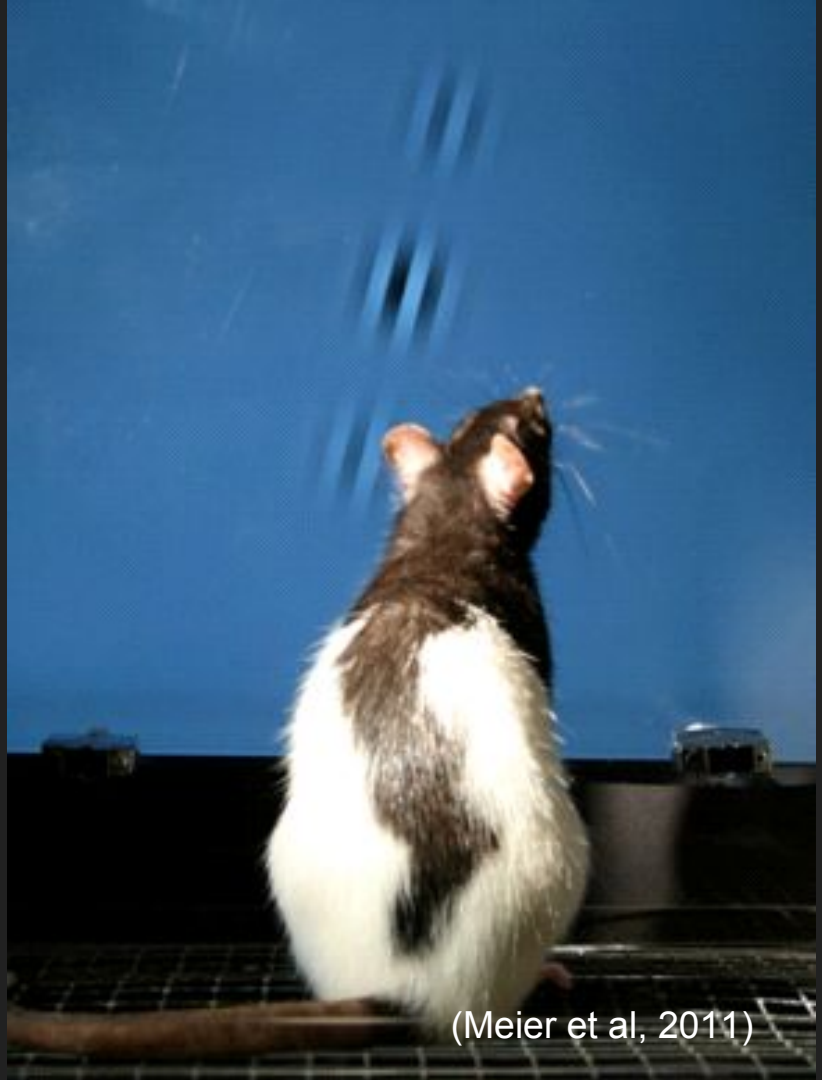


Sending Info to the Brain

- We need to link events that occur in the world with the brain's activity.
- The means delivering sensory (visual, auditory, etc) stimuli to our subjects.

PsychoPy for Sensory Stimuli

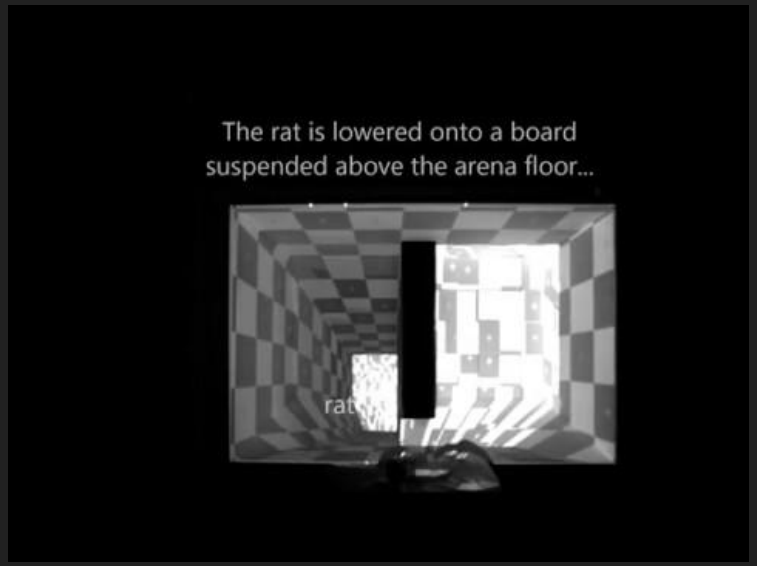
```
from psychopy import visual, event

window = visual.Window()
gabor = visual.GratingStim(pos=(0.5, 0.5))

while 'escape' not in event.getKeys():
    gabor.draw(window)
    window.flip()
```



(Meier et al, 2011)

3D Visual Stimuli

- Virtual Reality Research requires low-latency graphics.
- OpenGL-powered game engines like Pyglet make this possible in Python.



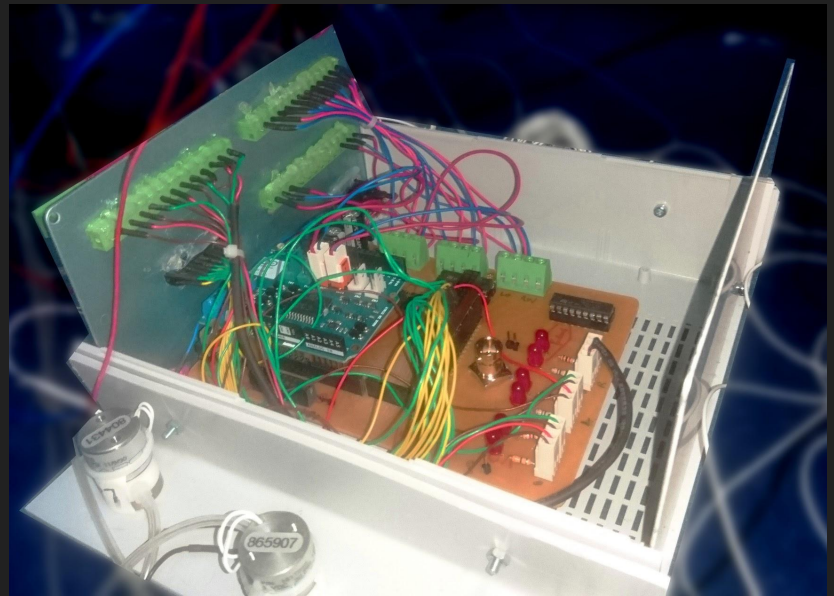
Pyglet and **ratCAVE**

```
import pyglet
import ratcave as rc

reader = rc.WavefrontReader('exp.obj')
sphere = reader.get_mesh('Sphere')

cam = rc.Camera(position=(1, 0, 0))
scene = rc.Scene([sphere], camera=cam)

win = pyglet.window.Window()

@win.event
def on_draw():
    win.clear()
    scene.draw()

pyglet.app.run()
```

http://github.com/neuroneuro15/ratcave

Animal Training

- We teach our rats to perform various tasks.
- The rats are rewarded.



Python-Controlled Arduinos

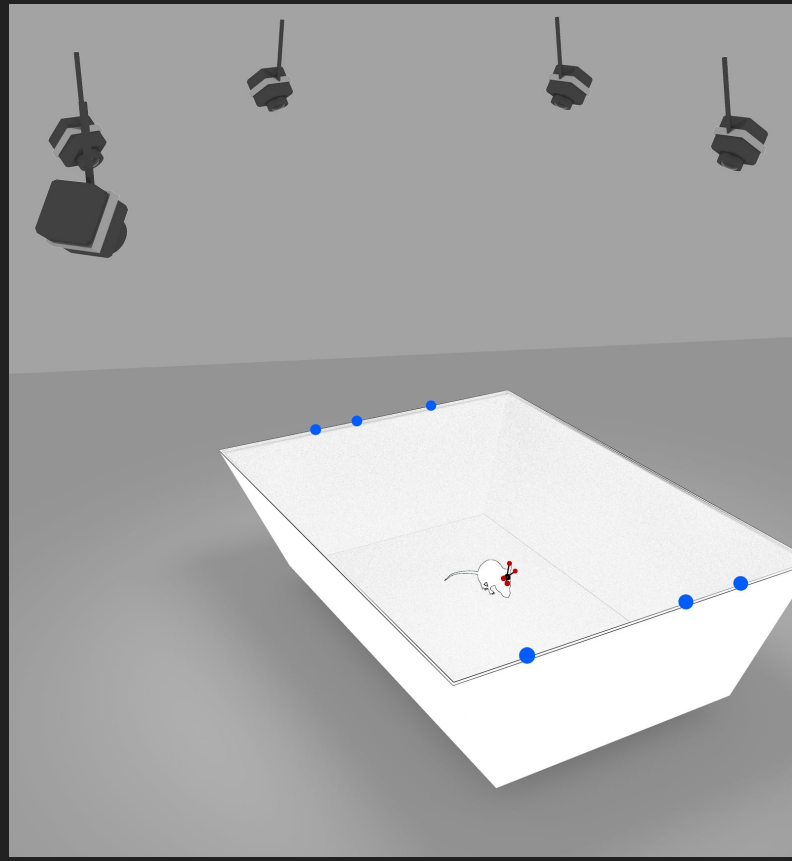
```
from serial import Serial

arduino = Serial('/dev/ttyACM0')
arduino.write('RightWater,3')

data = arduino.readline()
```

Networking Together Our Devices

- We need to synchronize many devices together during our experiments.
- Let's connect to it over the network!

Socket for network connections,
struct for splitting bytestrings into variables

```
from socket import socket, AF_INET, SOCK_DGRAM
import struct

sock = socket(AF_INET, SOCK_DGRAM, 0)
sock.bind('192.12.0.0')

message = sock.recv(128)
x, y, z = struct.unpack('3f', message)
```



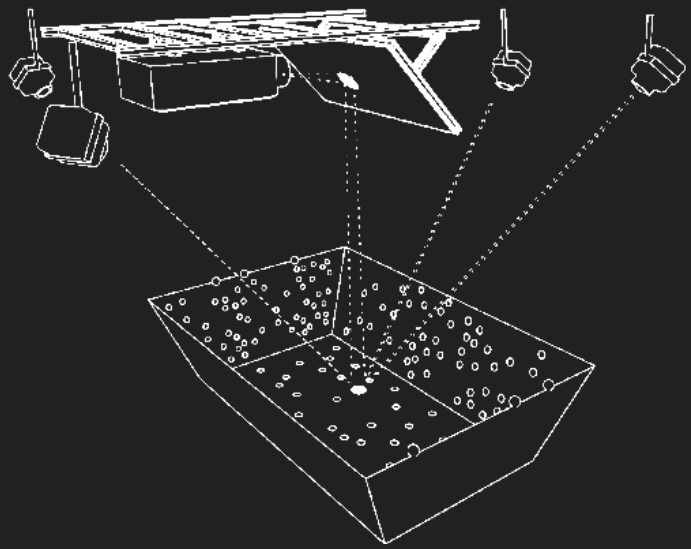
Refine Control of Hardware

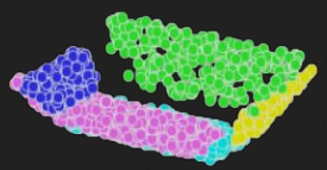
- We need to connect to commercial hardware

```
from vpixxlib import ProPixx

projector = ProPixx()
projector.setSleepMode(False)
projector.setLedIntensity('100.0')
```

Cython for Interfacing to C Libraries

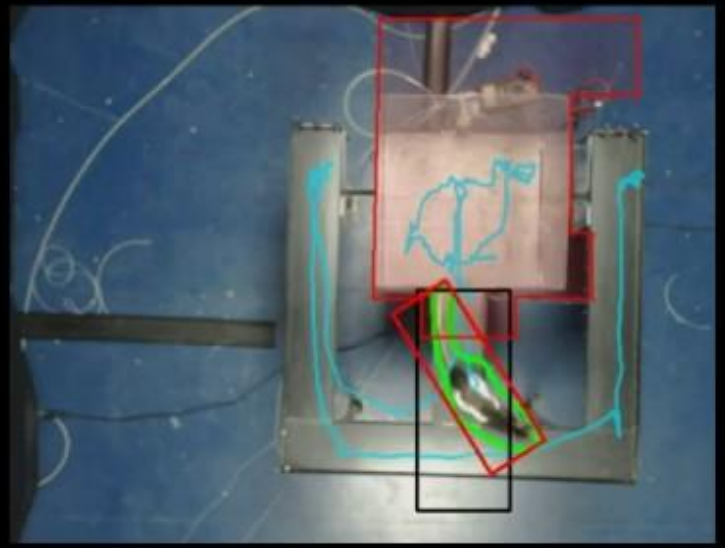
```
cdef extern from "NPTrackingTools.h":
    TT_Initialize()

def initialize():
    """Initializes camera connection."""
    return TT_Initialize()
```





Motion Tracking

- We need to link brain activity with the location of the subjects in the maze.
- We use a webcam-based tracking system

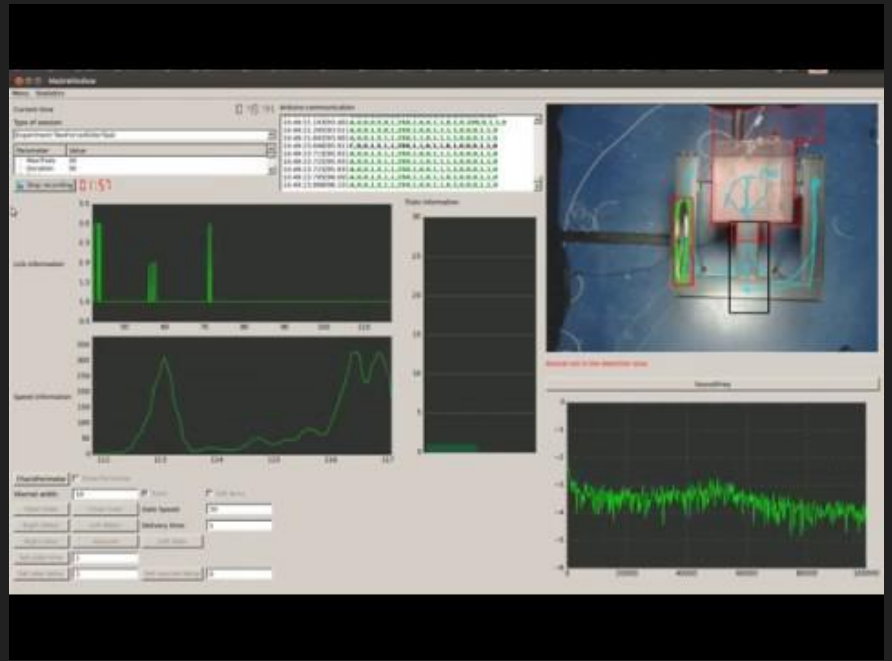


OpenCV for Computer Vision

```
import cv2 as cv

webcam = cv.VideoCapture(0)
_ ,img = webcam.read()

img = cv.createBackgroundSubtractorKNN(50000,400.0,True).apply(img)
img = cv.GaussianBlur(img, (11, 11), 0)
img = cv.threshold(img, 100, 255, cvv.THRESH_BINARY)[-1]
contours = cv.findContours(img, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_SIMPLE)[-2]
```

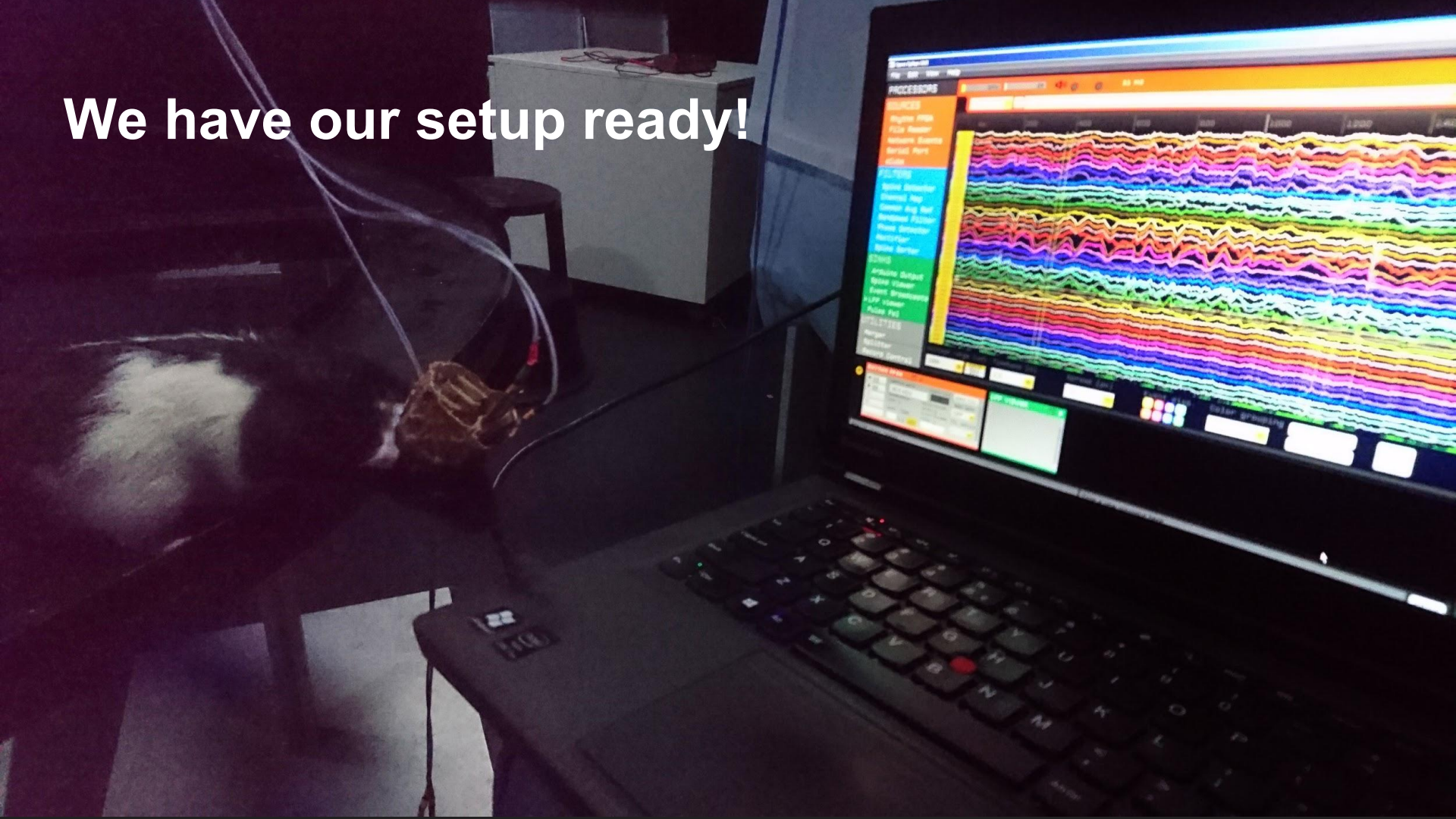
Graphic User Interfaces

- We can make user-friendly graphics interfaces
- Flexible way to standardize procedures

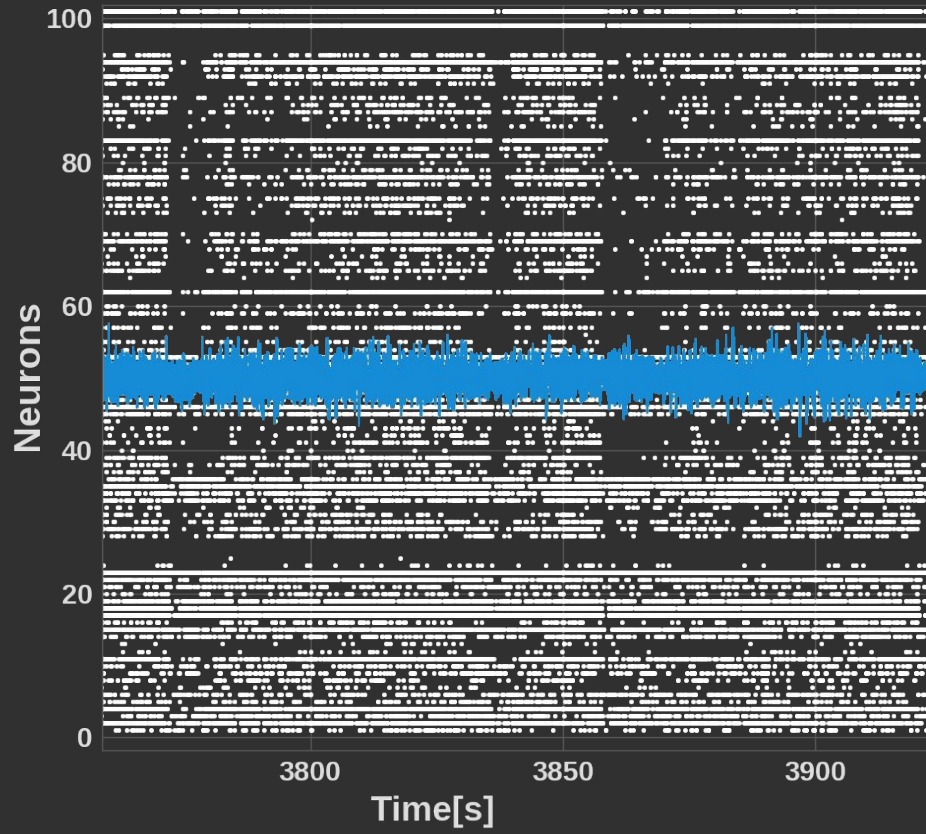


PyQt provide wide range of visual interfaces

```
from PyQt5.QtWidgets import
QApplication,
       QMainWindow, QPushButton

app = QApplication([])
win = QMainWindow()

button1 = QPushButton('Run Exp.', win)
button2 = QPushButton('Stop Exp.', win)
button2.move(0, 30)

win.show()
app.exec_()
```


We have our setup ready!

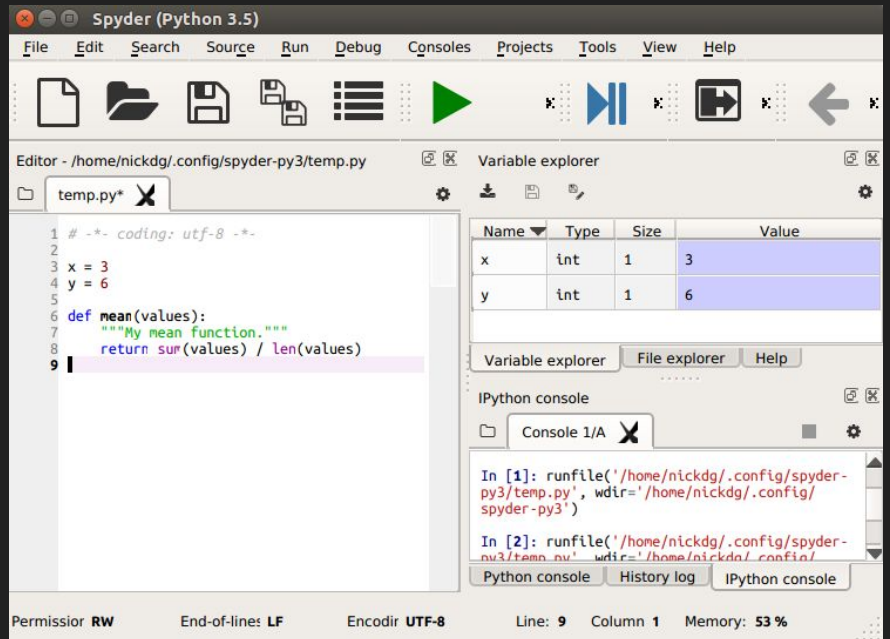
Analyzing Data



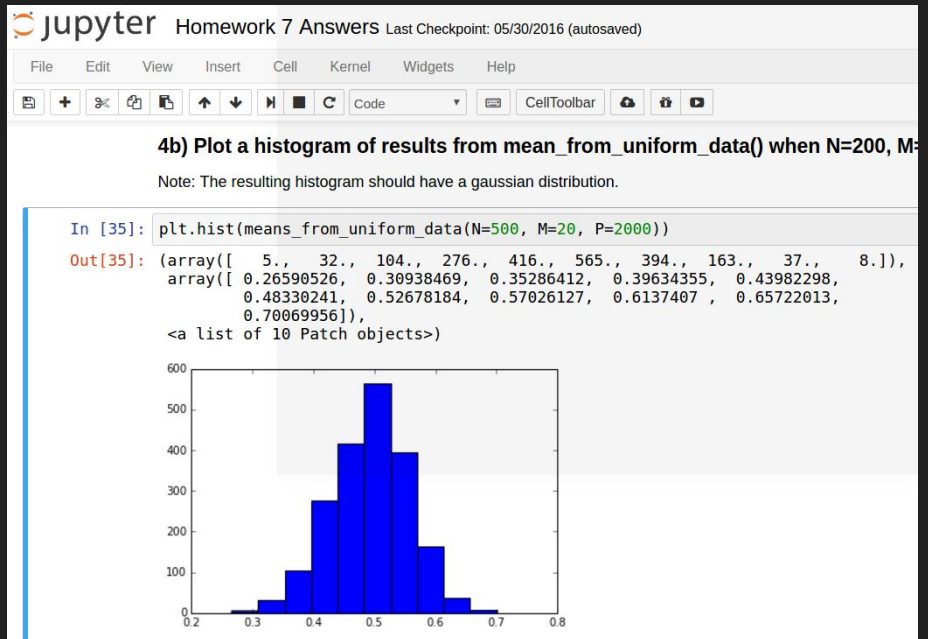
Interactive Data Analysis

- While programming, we want to continuously explore our data, view plots, and share our analyses with colleagues. Editors with **IPython** make these workflows possible.

Spyder



Jupyter



Reading Data from Files

Open Ephys Electrophysiology Data

```
import OpenEphys as oe

chan1 = oe.loadContinuous('CH1.continuous')
data = oe.loadFolderToArray(‘myDataFolder’)
```

Pickle for Python objects

```
import pickle

with open(‘data.pkl’) as f:
    data = pickle.load(f)
```

Pandas for reading Tables in many formats

```
import pandas as pd

df = pd.read_csv(‘mydata.csv’)
df = pd.read_html(‘mydata.html’)
df = pd.read_hdf(‘mydata.h5’, ‘spikes’)
```

Pandas and **SQLAlchemy** for databases

```
import pandas as pd
from sqlalchemy import create_engine

eng = create_engine(‘sqlite:///data.db’)
conn = eng.connect()
df = pd.read_sql_table(‘spikes’, conn)
```

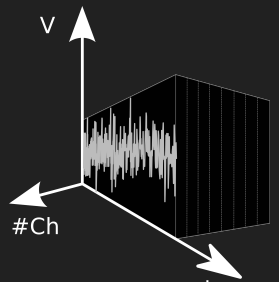
Powerful Computation and Data Manipulation

Numpy for arrays and N-dimensional matrices

```
import numpy as np

dat = np.random.random(200000)
dat_squared = dat ** 2

dat_matrix = dat.reshape(200, 1000)
dat_mean = dat.mean(axis=1)

[U,D,V]=np.linalg.svd(np.cov(f_spk))
```

Pandas for tables and time series

```
import pandas as pd

df = pd.read_csv('ratdata1.csv')
x_pos = df['x']

df_smoothed = df.rolling.mean()
df.head()
```



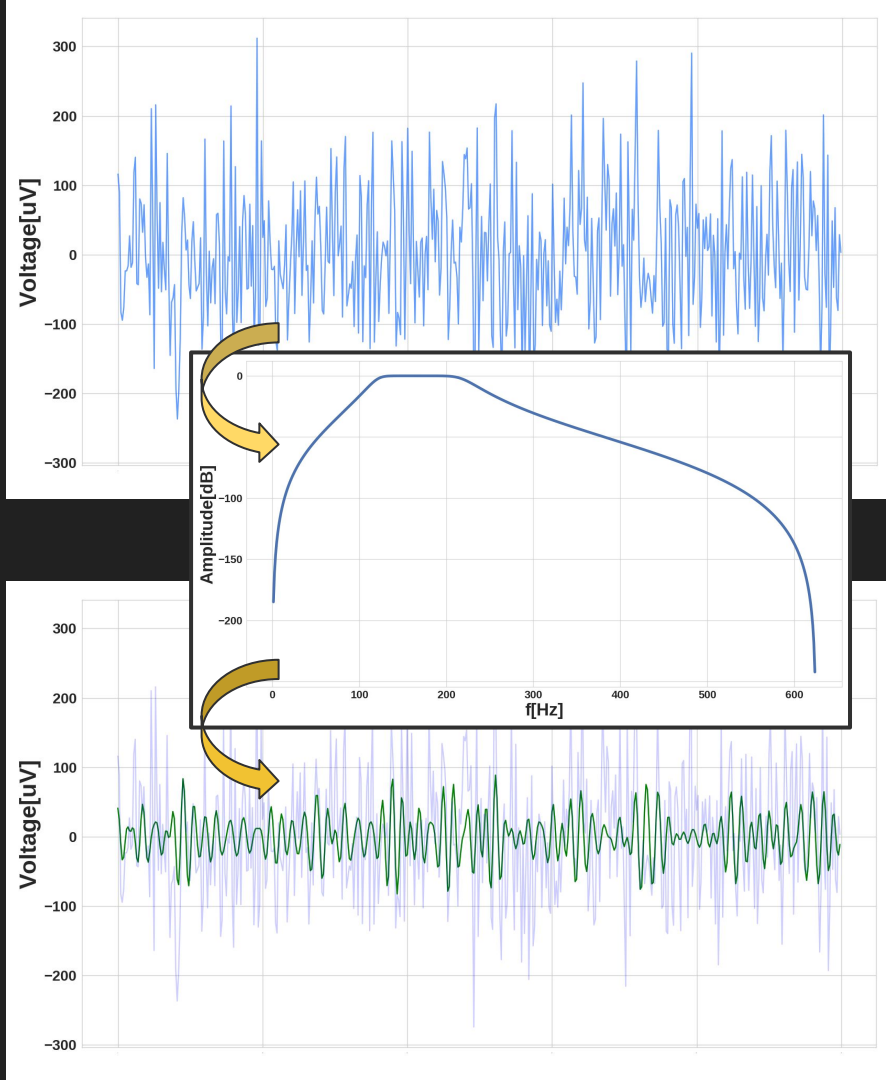
unit_id	Condition	X	Y	Session
1.0	Ctrl	-34.944474	-16.431694	1
5.0	Ctrl	-23.913915	15.398286	1
7.0	Ctrl	-18.440997	29.290983	1
9.0	Ctrl	-40.939740	-14.959865	1
16.0	Ctrl	-3.729383	-3.559295	1

Signal Processing

- Analysis of the spectral content reveals essential features of the neural code

Scipy-Signal for signal processing

```
from scipy import signal
import numpy as np

freqs = np.array([120., 220.])

filter = signal.butter(N=4,
   Wn=freqs / samp_rate * 2,
   btype='bandpass')

filtered_signal = signal.filtfilt(filter[0],
    filter[1], x=signal)
```

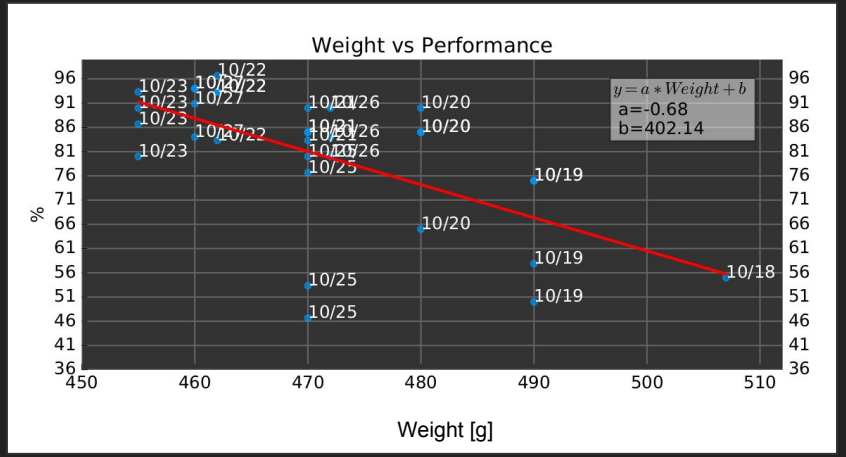


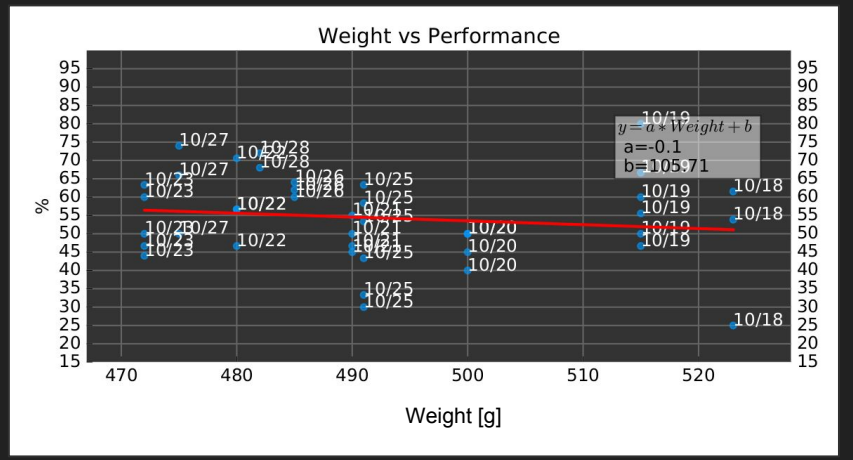
Machine Learning

- Supervised and unsupervised machine learning algorithms

Scikit-learn for extracting more features

```
from sklearn import linear_model

weight_train = dat[:-25,0]
weight_test = dat[-25:]

performance_train = dat[:-25]
performance_test = dat[-25:]

regr = linear_model.LinearRegression()

regr.fit(weight_train, performance_train)
performance_regr = regr.predict(weight_test)
```





Statistics

- Do our experimental observations fit a pattern?
- Could it have simply arisen from chance?

Scipy-Stats for hypothesis testing and distributions

```
from scipy import stats

gaussian_dist = stats.normal(0, 1)
rand_dat = gaussian_dist.rvs(200)

c1 = [1, 2, 1, 2, 1]
c2 = [5, 6, 7, 5, 6]
res, p = stats.ttest_ind(c1, c2)
```

StatsModels for building and testing linear models

```
import statsmodels.api as sm
formula = 'is_shifted ~ dX * dY * Session'
fit = sm.Probit.from_formula(formula, data=df)
print(fit.summary2())
```

```
                          Results: Probit
==================================================================
Model:              Probit           Pseudo R-squared:  0.322
Dependent Variable: is_Shifted       AIC:               107.3049
Date:               2016-09-23 11:44 BIC:               128.1463
No. Observations:   100              Log-Likelihood:    -45.652
Df Model:           7                LL-Null:           -67.301
Df Residuals:       92               LLR p-value:       2.9223e-07
Converged:          1.0000           Scale:             1.0000
No. Iterations:     7.0000
------------------------------------------------------------------
                     Coef.  Std.Err.    z    P>|z|   [0.025  0.975]
------------------------------------------------------------------
Intercept           -0.1511   0.2120 -0.7126 0.4761 -0.5665  0.2644
dX                  -0.1501   0.0377 -3.9804 0.0001 -0.2241 -0.0762
dY                  -0.0266   0.0207 -1.2854 0.1986 -0.0671  0.0140
dX:dY               -0.0094   0.0035 -2.6801 0.0074 -0.0162 -0.0025
np.exp(Session)      0.0035   0.0013  2.6672 0.0076  0.0009  0.0060
```

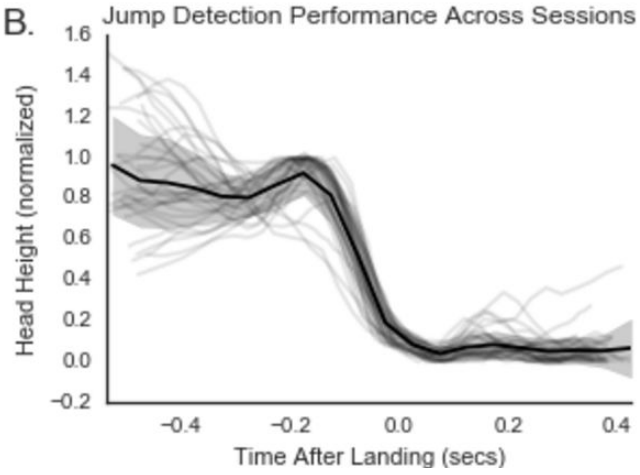
Data Visualization

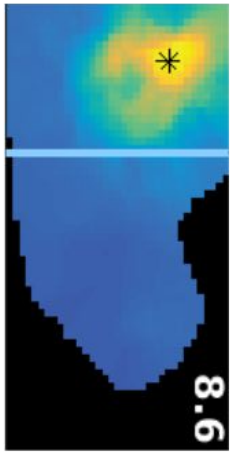
Matplotlib for fully-customizable charts

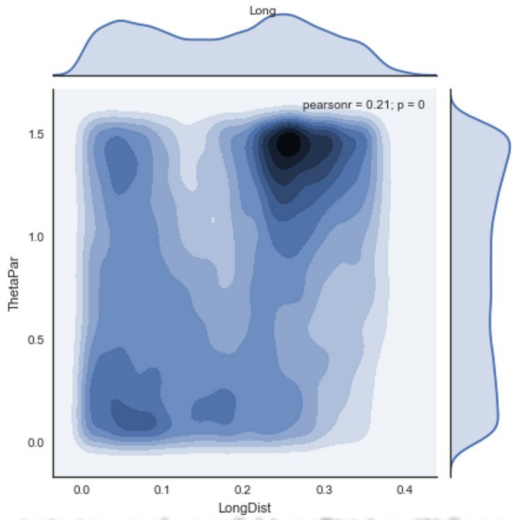
```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3], [8, 10, 2], ‘r’)
plt.xlabel(‘My X Axis’)

plt.show()
```

Seaborn for quickly building matplotlib charts

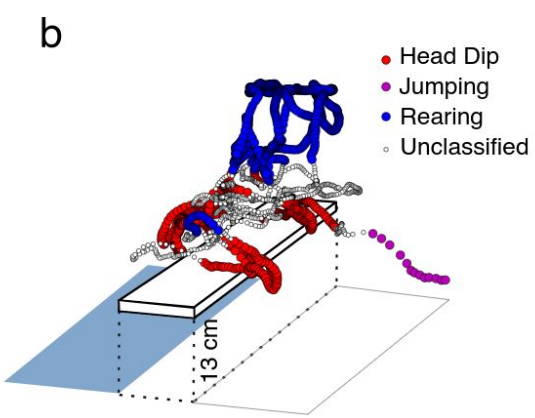
```
import seaborn as sns

sns.jointplot(x=’Size’, y=’Speed’, data=df)
```

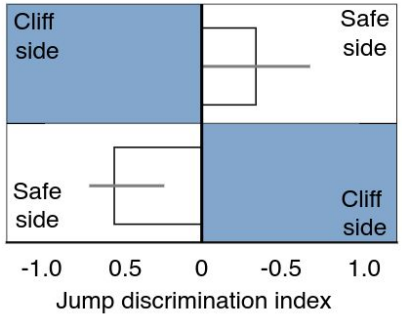


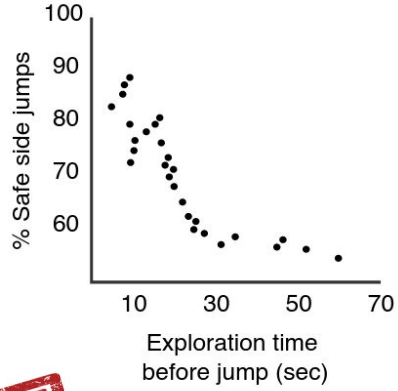




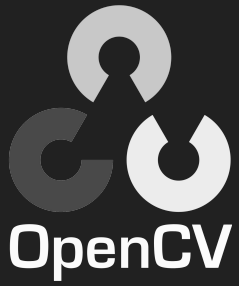
… in order to discover things about the brain (and publish!)



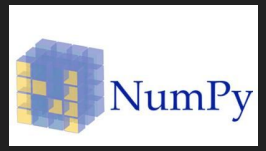




Python has rich ecosystem of packages for neuroscientists.


OpenCV


SM


NumPy

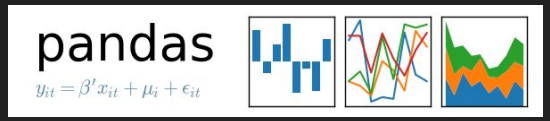

Qt


Cython


Seaborn


jupyter


matplotlib


pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$


scikit
learn


PY
SERIAL

Why does Python have so many resources for scientists?


It’s the community.

Acknowledgments

- Prof. Anton Sirota
- CNS Department
- RTG, Synergy
- LMU Munich
- PyData

Graduate School of Systemic Neurosciences
LMU Munich

SyNerg
Munich Cluster for Systems Neurology

LMU
LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

PyData
Barcelona 2017

Reference List and Request for Questions

- RatCave. http://github.com/neuroneuro15/ratcave
- Open Ephys. https://github.com/open-ephys. Accepted manuscript ‘Open Ephys: An open-source, plugin-based platform for multichannel electrophysiology’, Joshua Handman Siegle, Aarón Cuevas López, Yogi Patel, Kirill Abramov, Shay Ohayon and Jakob Voigts

- OpenCV.
  - Here the documentation for background substraction. http://docs.opencv.org/trunk/de/de1/group__video__motion.html
  - Here tutorials covering different aspects of OpenCV https://opencv-python-tutroals.readthedocs.io/en/latest/index.html
  - Here a great professional of image processing http://www.pyimagesearch.com/

- PySerial. Complete documentation of the project. https://pythonhosted.org/pyserial/

- PyQt.
  - Documentation for PyQt4 http://pyqt.sourceforge.net/Docs/PyQt4/
  - Documentation for the original Qt project in C++. Despite of that it is a great source of information of the properties of the objects that the python bindings can provide. http://doc.qt.io/qt-4.8/
  - A recommended book. “*Rapid GUI Programming with Python and Qt: The Definitive Guide to PyQt Programming”*,Mark Summerfield

Javier Martinez Alcantara
@jmaralc

Nicholas Del Grosso
@neuroneuro15

Spare Slides

Neuroscience is a Broad Field Containing Many Disciplines