# Processing Calcium Signaling Fluorescence Microscopy Image Stacks


*John Rugis (University of Auckland)*
*James Sneyd (University of Auckland)*
*David Yule (University of Rochester)*



Funded by

# The value of interdisciplinary collaboration

**In our case,**

*John* - ***Computer Science***
*James* - ***Mathematics***
*David* - ***Physiology***

**NEW:** Enhanced computation for model building, simulation and visualisation.

***Broad coverage was probably why the project was awarded the latest grant!***

Support from NIH (NIDCR R01- DE14756, DE19245)

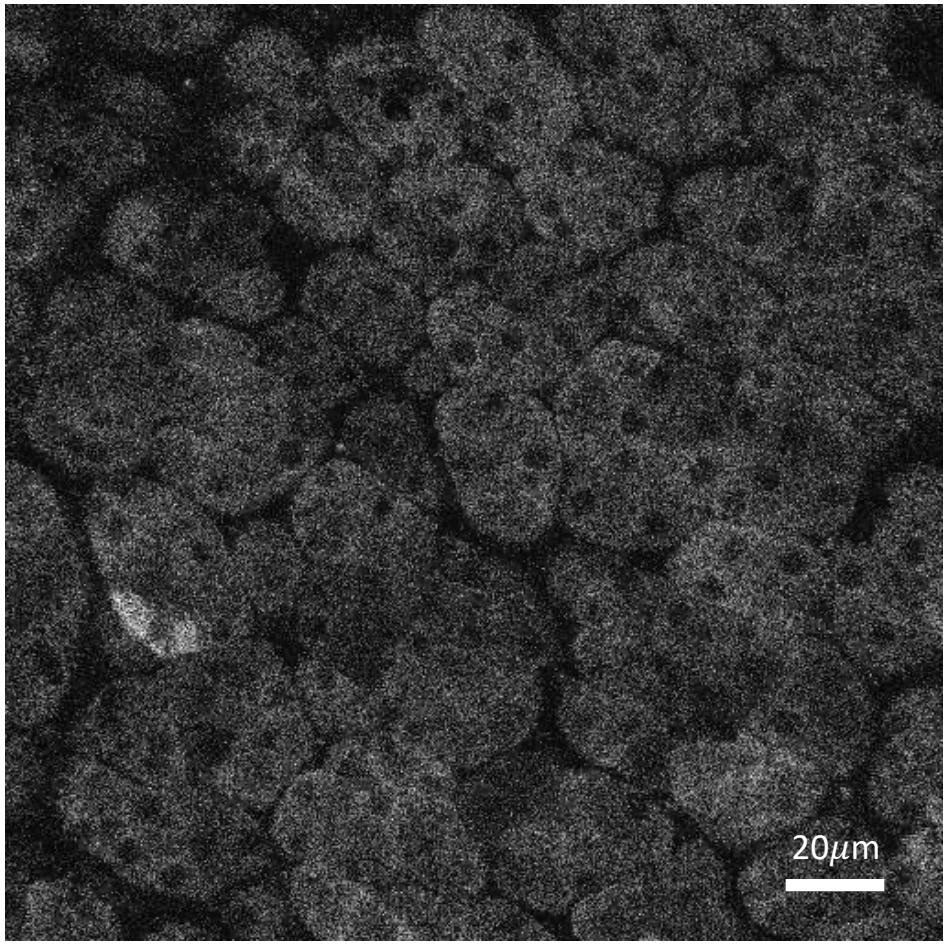
# Some (biology) background…

- Calcium signaling occurs within the cells of the parotid gland which is responsible for the production of saliva.
- David's lab has recently acquired *in-vivo* microscopy capabilities. Fully intact live cells in functional gland!
- Extracted information used for direct analysis and in building mathematical models for simulation.

But in this presentation, image stack processing only.

# Microscopy image stacks

A typical unprocessed image stack:




Calcium fluoresces in regions-of-interest.

512 x 512 pixels per image,
up to 20 images per second,
for 30 or more seconds.

Each experiment run typically produces 4 image stacks.

**Problem:**

*The image stacks from a single experiment easily consume over 1GB of storage and analysis of the data from a single experiment took several weeks.*

## Goal:

*A custom computational workflow designed to semi-automate region-of-interest determination, data processing and analysis.*

## Software toolset

- A collection of Jupyter lab notebooks
- Python (computation) and Matplotlib (visualisation)
- Python scientific code libraries (extensive!)

# Lab notebook organisation

## Pre-processing

- Image stabilisation

## Analysis

- Image noise reduction
- Region of interest determination
- Calcium plots (many!)
- Summary plots

## Post-processing

- Frequency analysis
- Peak counting
- Movie making

# **Jupyter notebooks:** *Code hiding, context sensitive GUI user interface*

## Python notebook to plot apical region response to stimulation.

Assumes folder directory structure:

```
IMAGING
  image_stacks
  notebooks
  results
```

Execute the code sequentially, one block at a time, using <shift-return>.

•••

### Select image stack file(s).

Select multiple images using command-click.
Only select multiple stacks that have the same pixel and depth dimensions!

•••

Image stack(s)

MistGcamp-2_4x(20hz)_0003.tif
Mistgcamp-3.tif
Mistgcamp-3_0001.tif
Mistgcamp-3_0002.tif
Mistgcamp-3_0003.tif

# **Notebooks:** *Code hiding, context sensitive GUI user interface*

## Python notebook to plot apical region response to stimulation.

Assumes folder directory structure:

```
IMAGING
  image_stacks
  notebooks
  results
```

Execute the code sequentially, one block at a time, using <shift-return>.

```
import csv
import datetime
import glob
import ipywidgets as widgets
import matplotlib.pyplot as plt
from matplotlib import gridspec
import matplotlib as mpl
import numpy as np
import os
from utils import remove_large_objects
from skimage import exposure
from skimage import io
from skimage.morphology import binary_erosion, binary_dilation
from skimage.morphology import remove_small_objects
from skimage.measure import label, regionprops
from skimage.util import img_as_float32
```

# **Analysis notebook:** *Region of interest determination*

**Set display and calculation parameters.**

Note: An LUT color reference can be found at: https://matplotlib.org/3.1.1/gallery/color/colormap_reference.html

•••

| | |
|---|---|
| Image stack for ROI creation | 5Hz |
| Image stack LUT | coolwarm |
| Stimulation start frame | 100 |
| Stimulation done frame | 250 |
| Image stack frames per second | 10 |
| Image data bits | 10 |
| Small region cull size | 16 |
| Number of dilations | 2 |
| Calculation method | average |
| ROI threashold quantile | 0.98 |
| Large ROI % to cull | 0 |

**Create a new time-stamped results directory.**

•••

# **Analysis notebook:** *Region of interest determination*

Create a new time-stamped results directory.

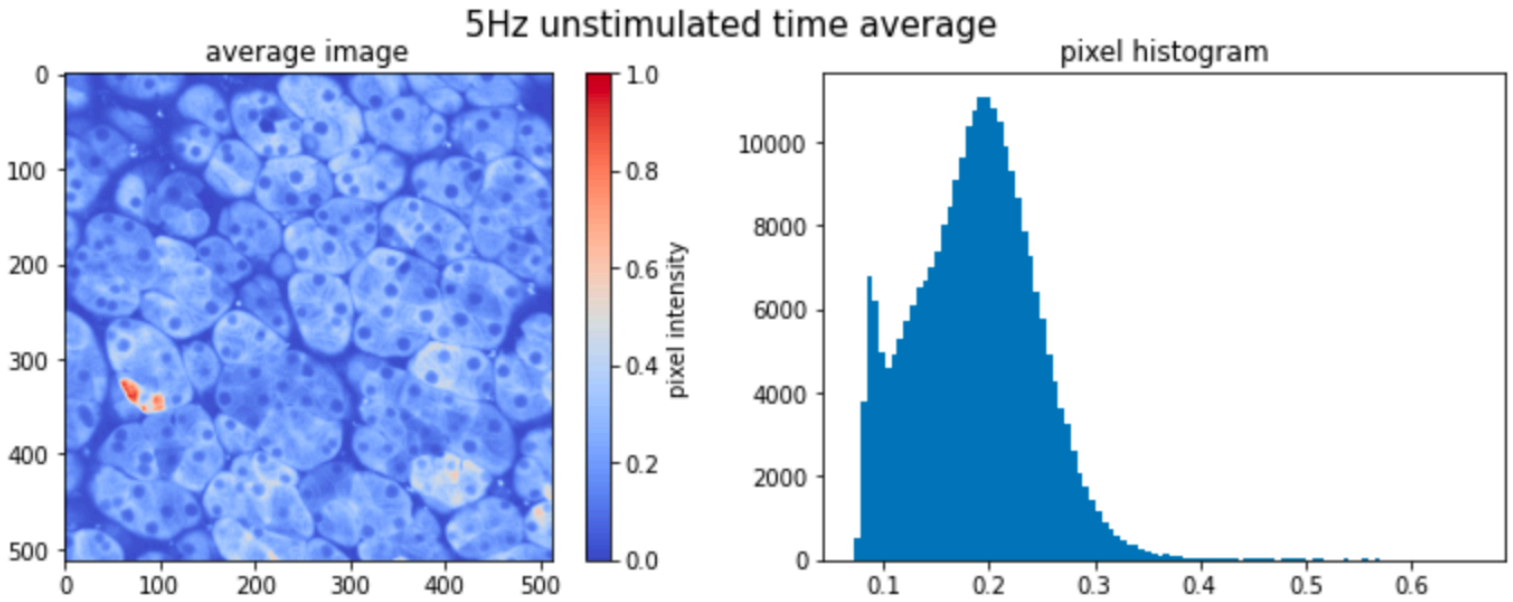
•••

Get the image stack for ROI creation.

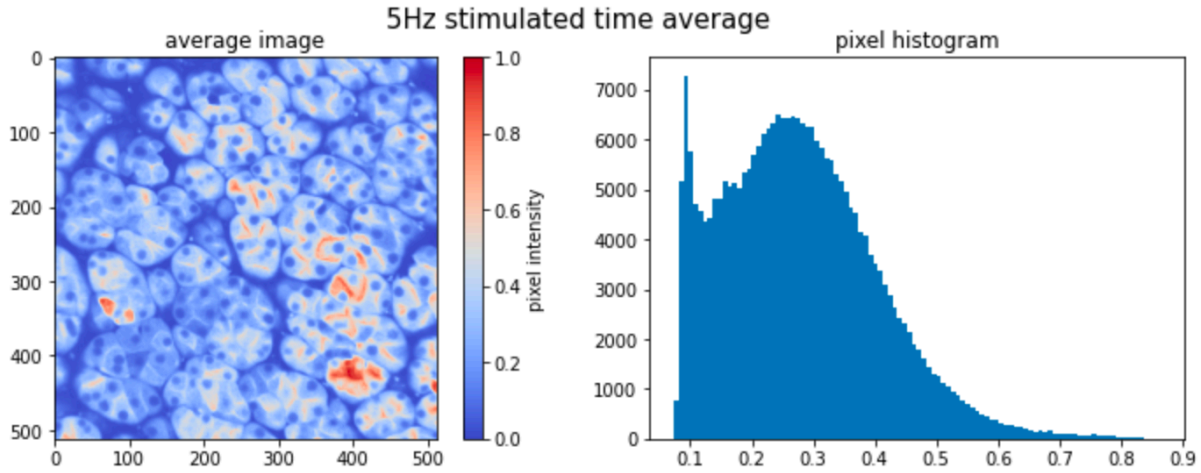
•••

Unstimulated average (or standard deviation) over time.

•••

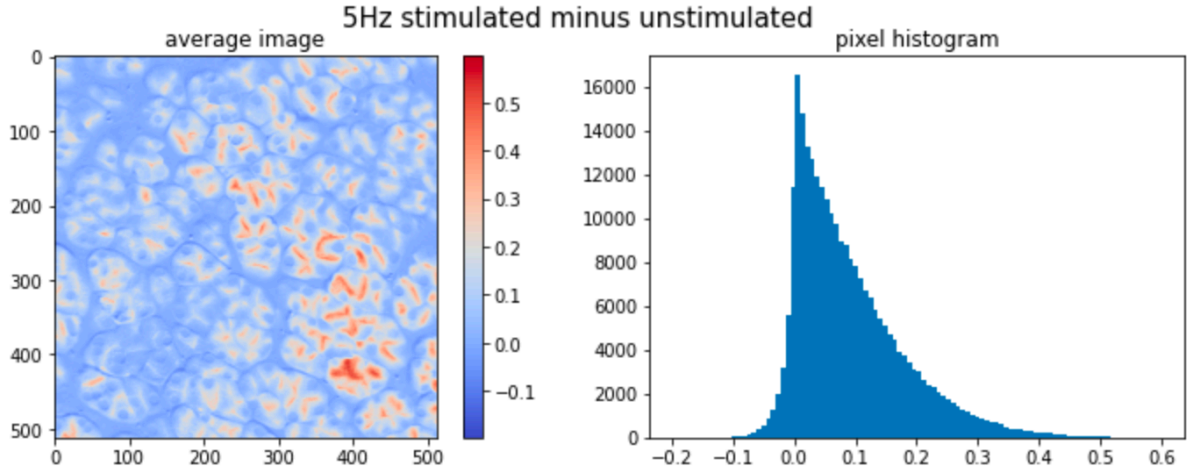


# **Analysis notebook:** *Region of interest determination*




Stimulated minus unstimulated average (or standard deviation) over time.

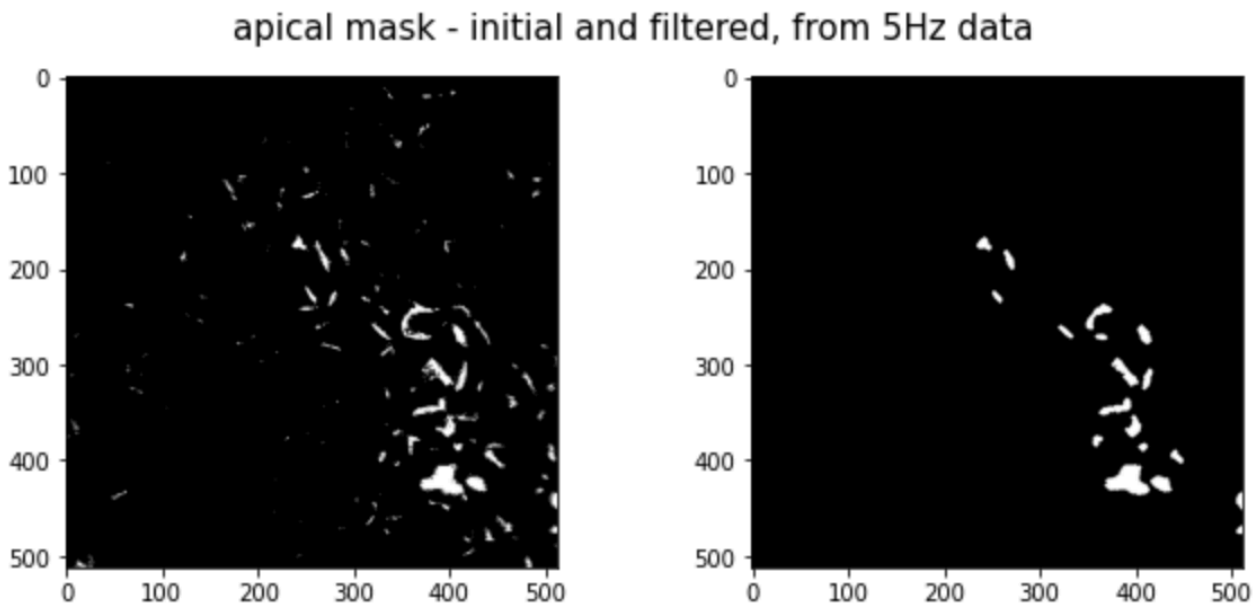
• • •



# **Analysis notebook:** *Region of interest determination*

Create apical region-of-interest mask.

•••




```
stimulation: 5Hz
number of apical regions identified:  18
apical ROI labels:  [ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18]
apical ROI pixel counts:  [138 130  72 255  94 183  65 302 142 267 215  87  61 105 889 254  87  43]
```

FOR REFERENCE: Annotate the apical mask regions by number.

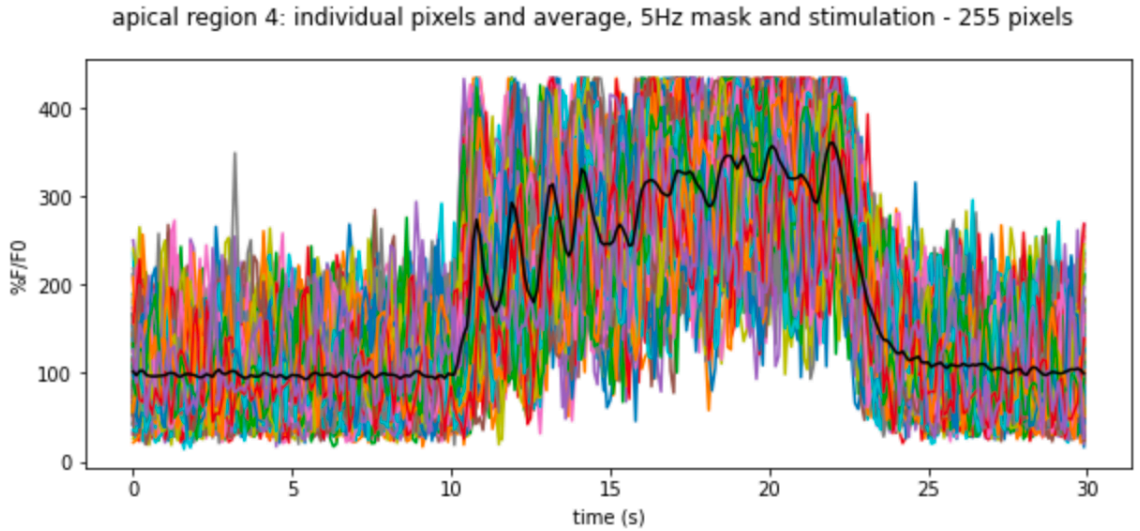
•••

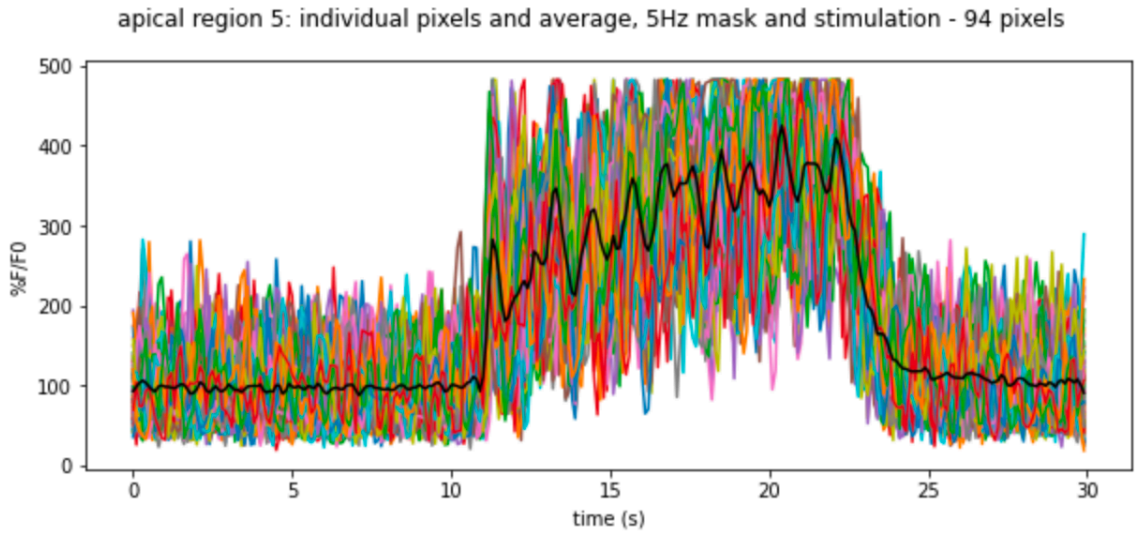
# **Analysis notebook:** *Region of interest determination*

Create apical region-of-interest mask.

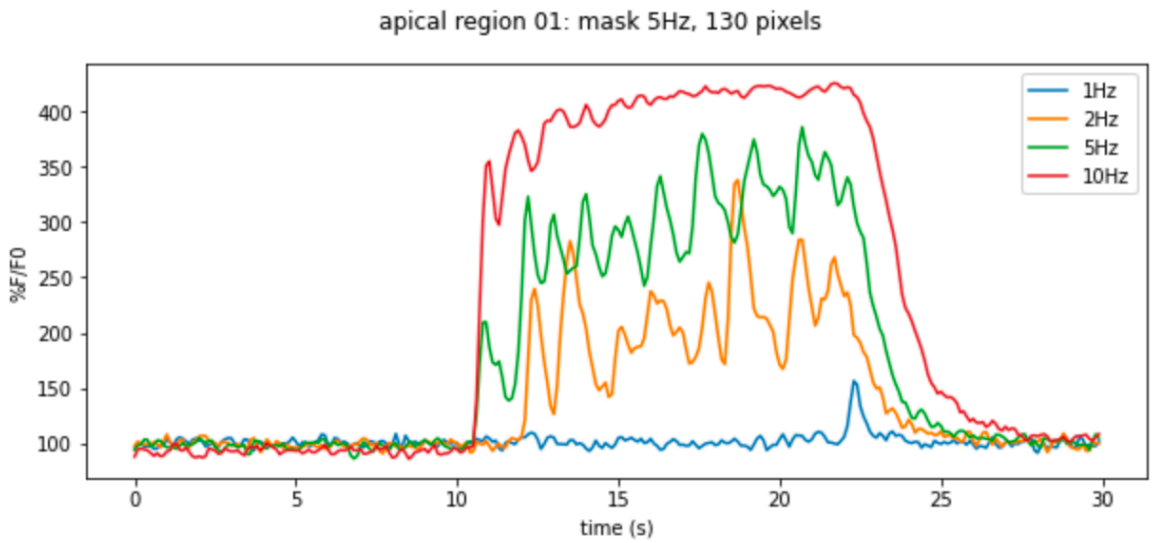
```
%matplotlib inline

os.system("rm -f " + resultsdir + "/apical_region*.*") # delete all exiting region files

fig, ax = plt.subplots(nrows=1, ncols=2, figsize = [10, 4])
fig.suptitle("apical mask - initial and filtered, from " + image_tags[roi_idx][0] + " data", fontsize=15)

# difference threashold filter
P = (O > np.quantile(O,roi_quantile)).astype(float)

# plot image
ax[0].imshow(P, norm=None, cmap='gray');
io.imsave(resultsdir + "/" + image_tags[roi_idx][0] + "-apical_mask_initial.png", 255*np.uint8(P), check_contrast=False)

# filtering: erosion, remove small, then dilation
Q = binary_erosion(P)
Q = remove_small_objects(Q, small_object)
for i in range(dilations):
  Q = binary_dilation(Q)
Q = remove_large_objects(Q, roi_cull)

ax[1].imshow(Q, norm=None, cmap='gray') # plot image
io.imsave(resultsdir + "/" + image_tags[roi_idx][0] + "-apical_mask_filtered.png", 255*np.uint8(Q), check_contrast=False)
plt.show()

# label and get pixel counts
R, n = label(Q, return_num=True)
io.imsave(resultsdir + "/" + image_tags[roi_idx][0] + "-apical_mask_labelled.tif", np.int16(R), check_contrast=False)
ROI, COUNT = np.unique(R, return_counts=True)
print("stimulation: " + image_tags[roi_idx][0])
print("number of apical regions identified: ", ROI.shape[0]-1)
print("apical ROI labels: ", ROI[1:])
print("apical ROI pixel counts: ", COUNT[1:])
print()
```

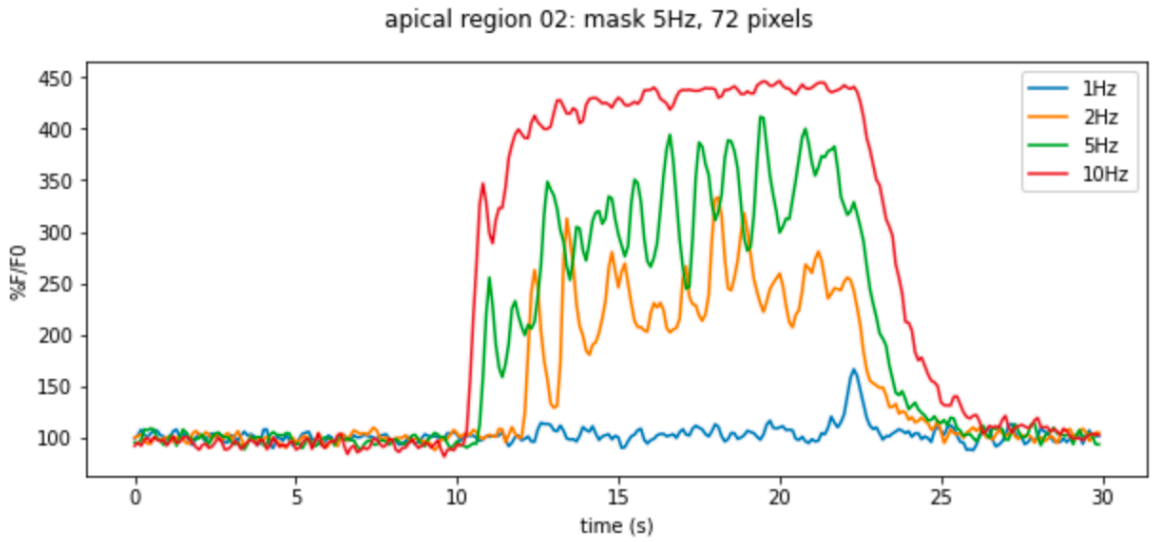
# **Analysis notebook:** *Calcium plots – all pixels & average per region*



# **Analysis notebook:** *Calcium plots – Summary per region*



# Post-processing notebook: *Frequency analysis*

## Python notebook for post-processing apical responses.

## Frequency analysis of results data.

Assumes folder directory structure:

```
IMAGING
  image_stacks
  notebooks
  results
```

Execute the code sequentially, one block at a time, using <shift-return>.

•••

### Select a results directory and set analysis frame range.

•••

Results dir
20200713-201826
20200805-160811
20200908-102432
20200908-103925

Selection OK.

Stimulation start frame 100

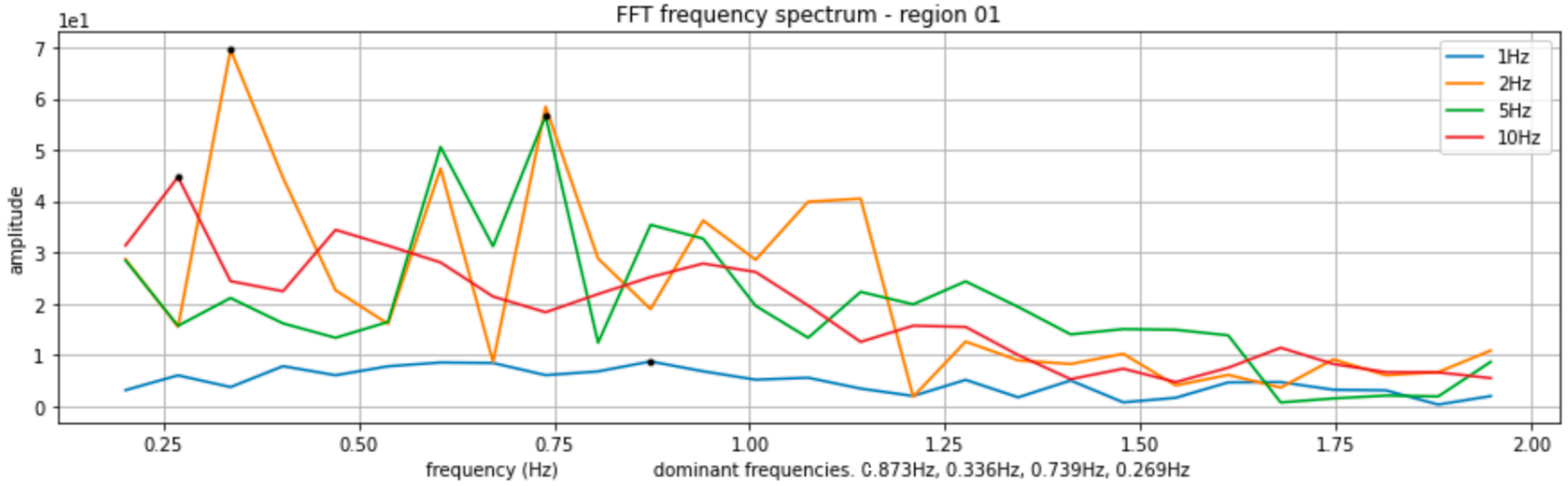
Stimulation done frame 250

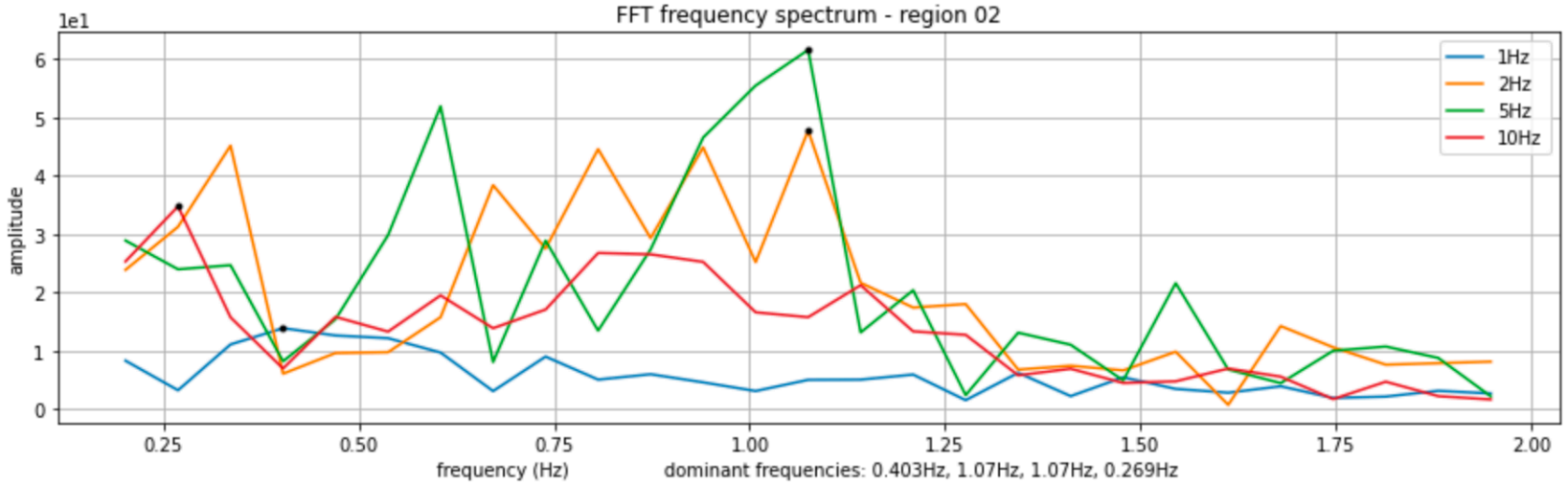
Calculation method FFT

Results scale linear

# **Post-processing notebook:** *Frequency analysis*

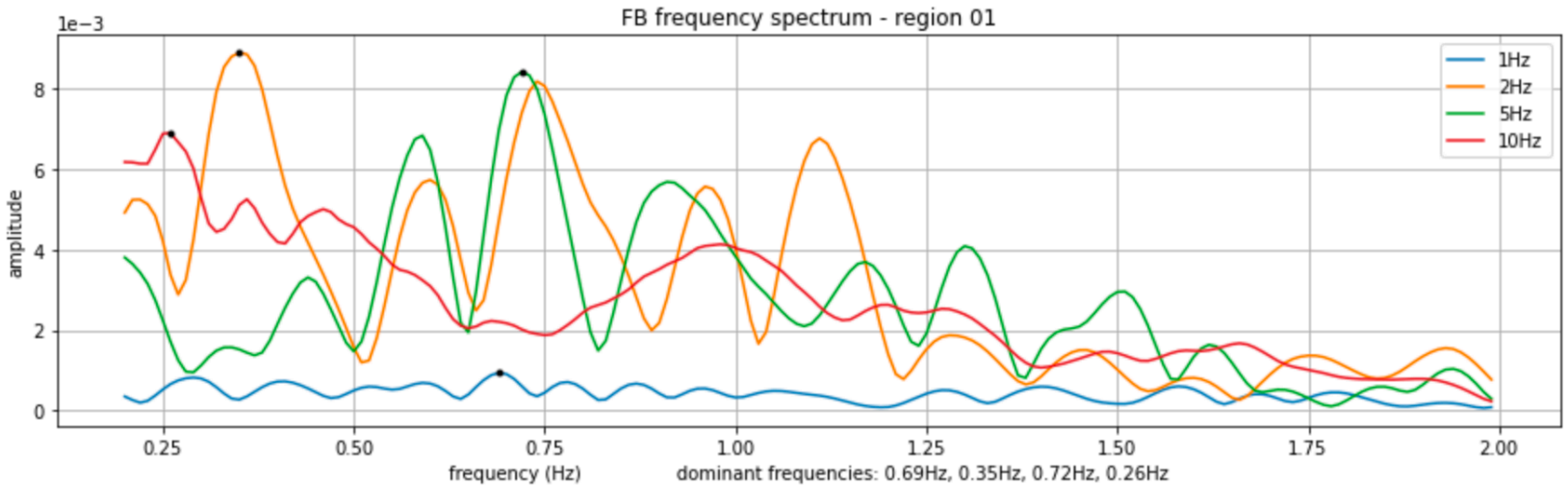
Frequency analysis of results over all regions.

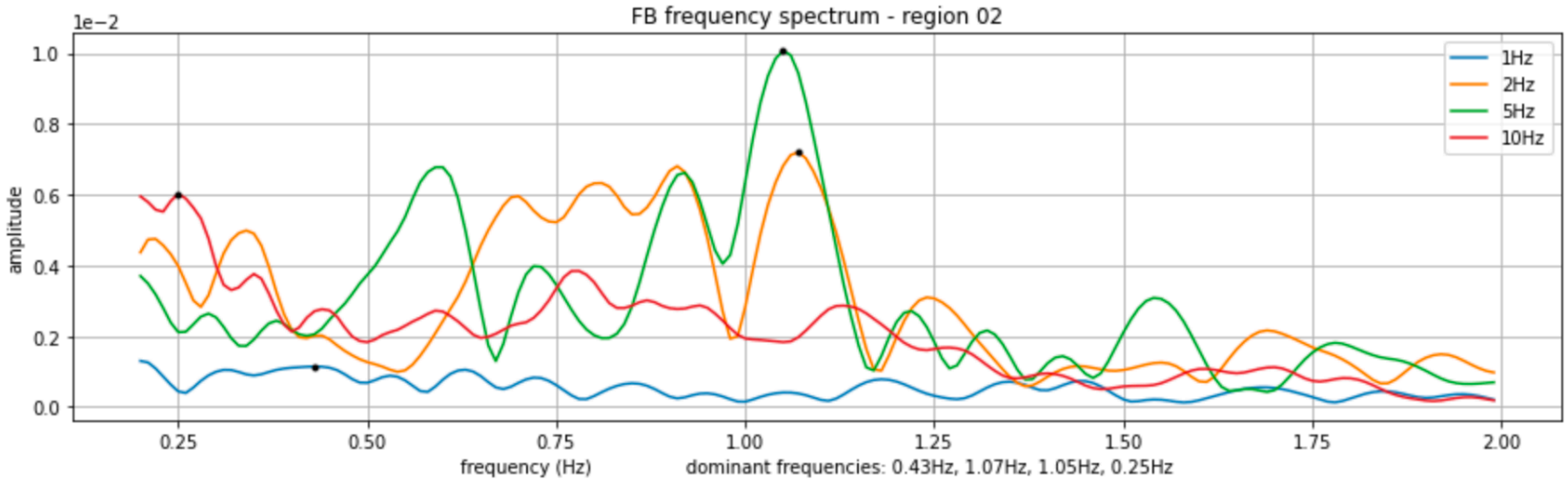


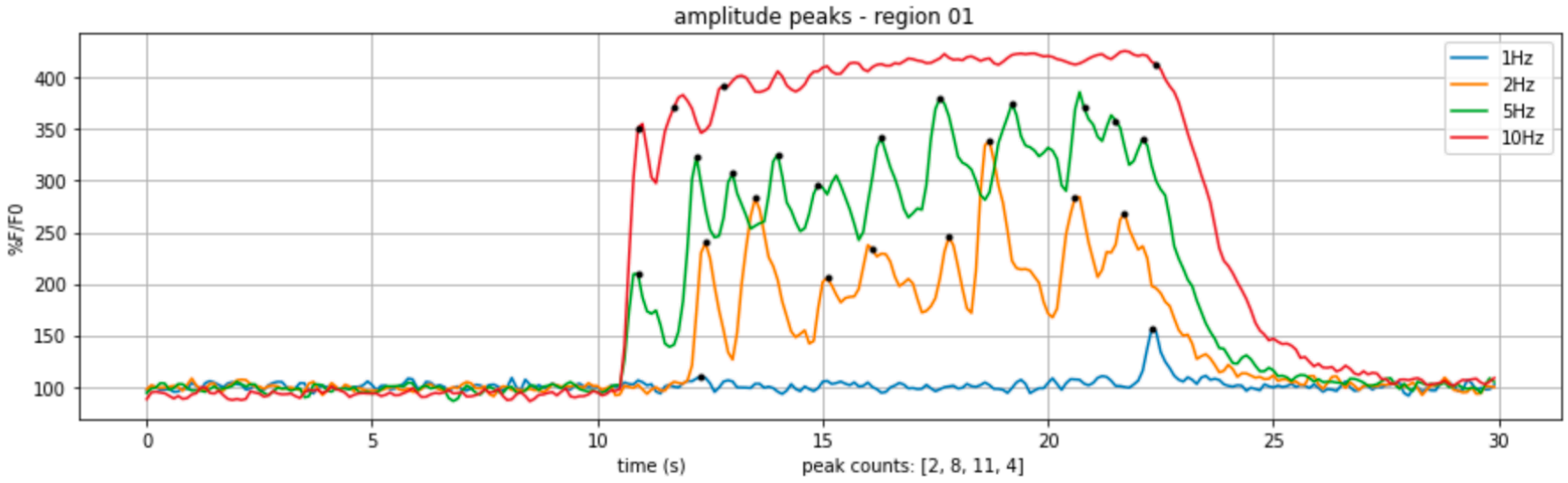
# **Post-processing notebook:** *Frequency analysis*

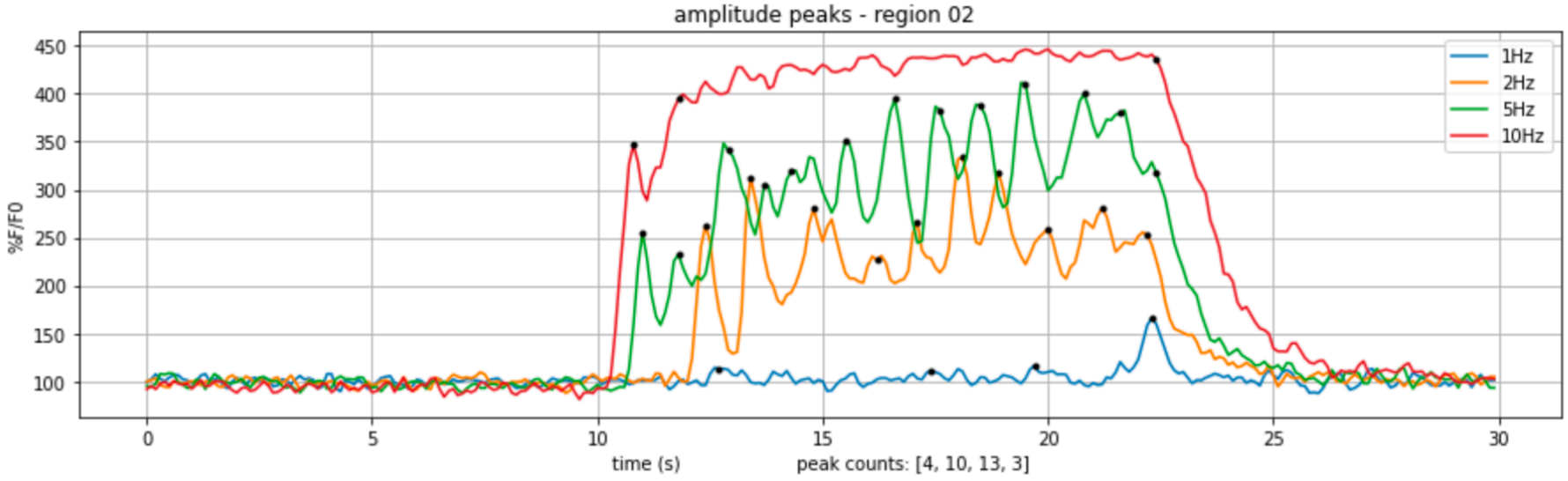
Frequency analysis of results over all regions.



# **Post-processing notebook:** *Peak counting*



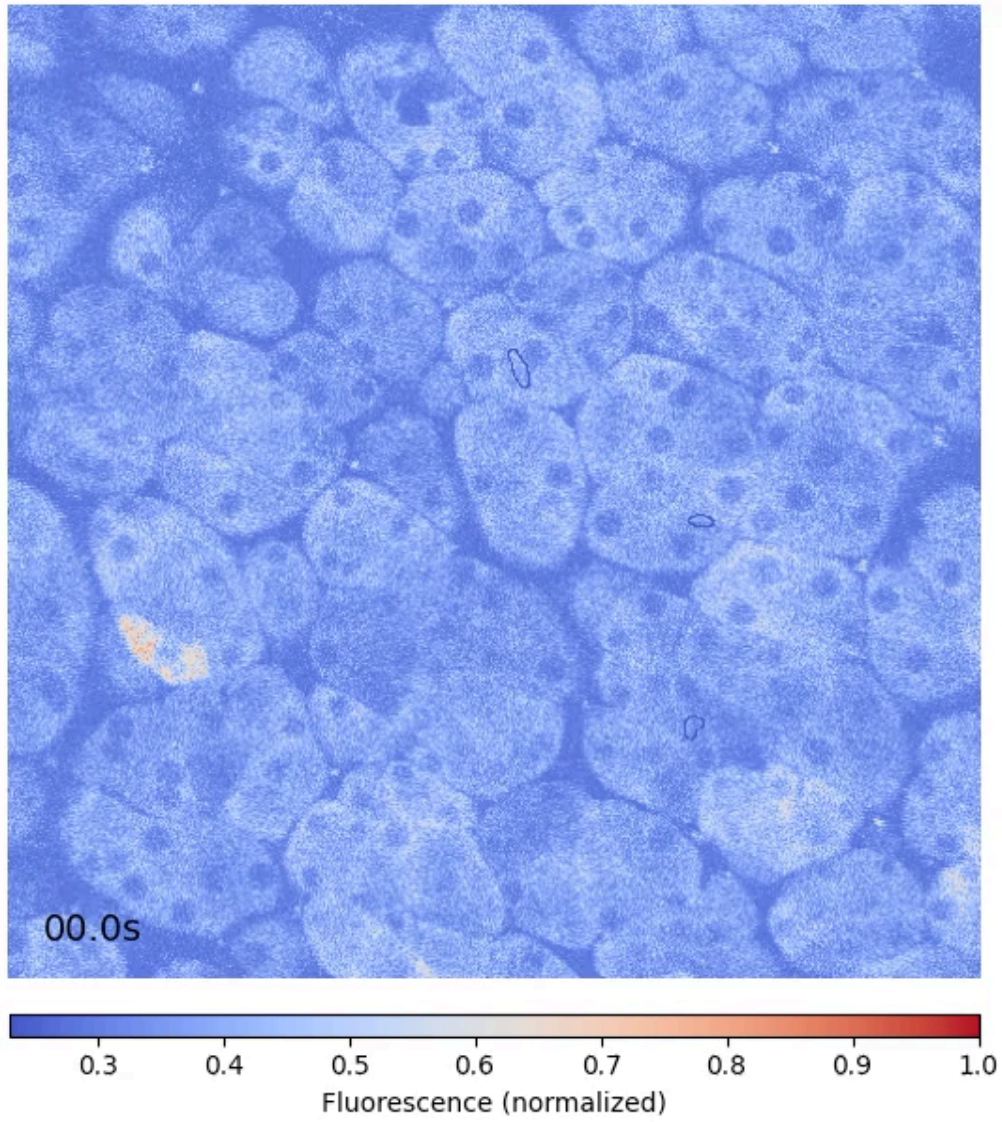



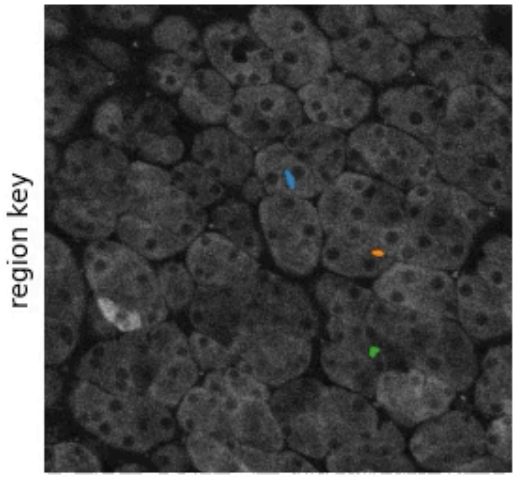
# Post-processing notebook: *Peak counting*

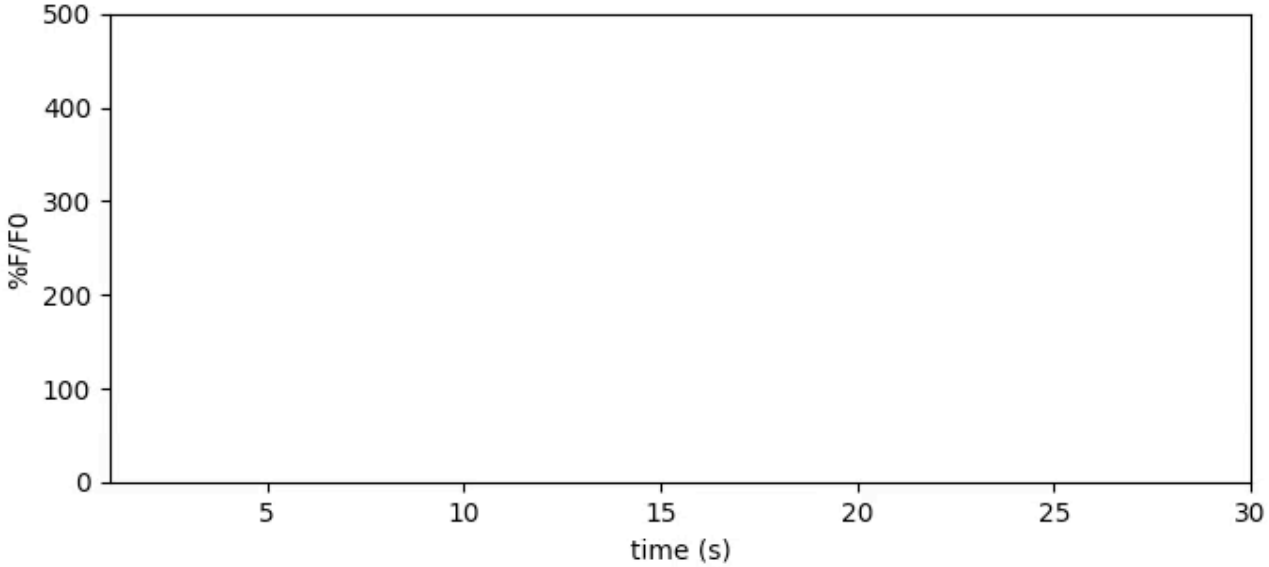
```
region = r.split('_')[-1].split('-')[0]  # get the region number
A0 = np.transpose(np.genfromtxt("../results/" + results_sel + "/" + r, delimiter=','))
A = A0[:,stim_start:stim_done]  # trim the data to the stimulated time
tmin = np.min(A[0])      # start time
tmax = np.max(A[0])      # finish time
trng = tmax-tmin         # time range
tstp = A0[0,1]-A0[0,0]   # time step
dmin = np.min(A[1:])     # minimum data value (over all traces)
dmax = np.max(A[1:])     # maximum data value
drng = dmax-dmin         # data range
X0 = A[0]                # the time axis
Y0 = (A[1:]-dmin) / drng # data axis, normalized to range(0, 1.0)

sr = p / trng   # the sample rate
pk = []
pts = []
for idx,y in enumerate(Y0): # for each trace
  f = interp1d(X0, y, kind='cubic')                    # define the resampling function
  X = np.linspace(tmin, tmax, p+1, endpoint=True)      # define the new time steps
  Y = f(X)                                             # resample the original signal

  # apply high-pass filter to eliminate the stimulation "bump" in the data
  sos = signal.butter(3, 0.1, btype='highpass', fs=sr, output='sos')
  Yf = signal.sosfiltfilt(sos, Y) # zero phase shift filter

  # apply low-pass filter to smooth out higher frequencies in the data
  sos = signal.butter(7, 2.0, btype='lowpass', fs=sr, output='sos')
  Yf = signal.sosfiltfilt(sos, Yf) # zero phase shift filter

  pks,_ = signal.find_peaks(Yf,prominence=0.04)        # find indices of peaks in the resampled, filtered data
  pk.append(len(pks))                                  # save the number of peaks
  pidx = np.around(stim_start + (stim_done-stim_start-1)*pks/p).astype(int)  # convert to indices in the original data
  pts.append([A0[0][pidx], A0[idx+1][pidx]])           # save the peaks as points in the original data
  ax.plot(A0[0],A0[idx+1],label=str(data_labels[idx]))       # plot the original data
  ax.plot(pts[-1][0],pts[-1][1],'k.')                        # plot the peak locations
```

# **Post-processing:** *Video clip*

$Ca^{2+}$ signals induced by neural stimulation



# Problem solved!

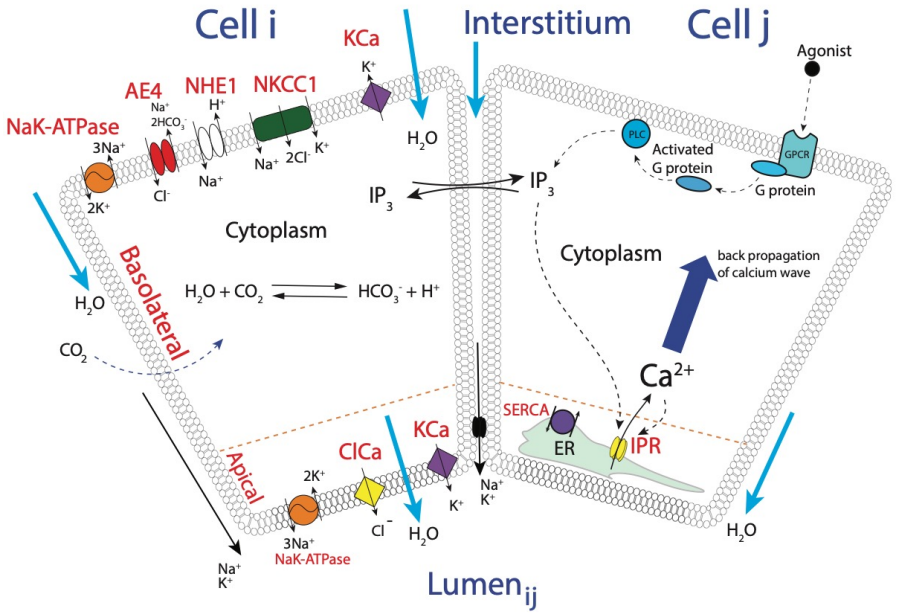
- *The notebooks are used by lab staff in a “cookbook” like fashion with very little training required.*
- *Initial analysis of an experiment can now be done in under twenty minutes.*
- *Naming conventions and file organisation structures reduce data handling complexity.*

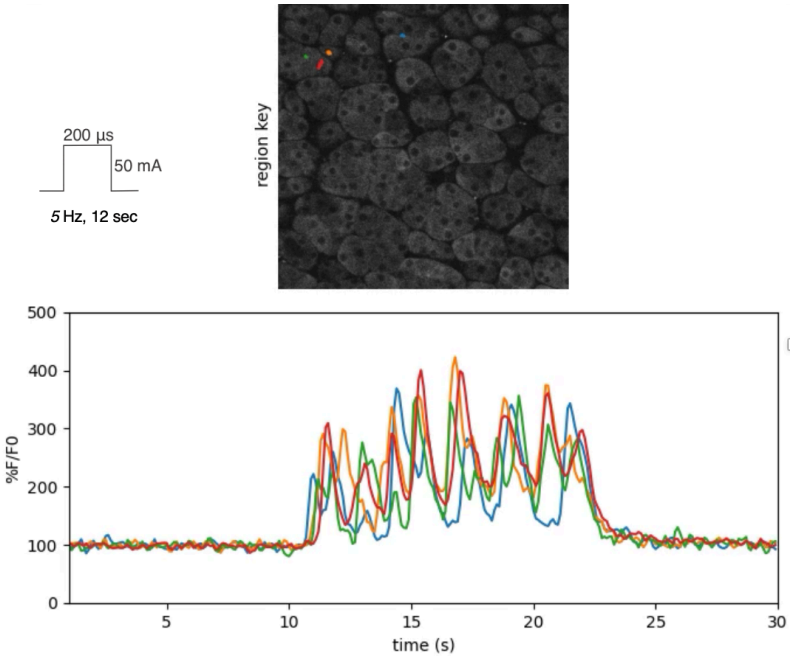
# Our research and Open Science

- *Our computational notebooks are publicly hosted on GitHub* under a GPL 3 license.*
- *Reproduceable results, transparency.*
- *Encouraged by our funding agency.*
- *It can be personally rewarding to share ideas!*

*** https://github.com/jrugis/plot_apical**

# In summary…




Physiology laboratory + Mathematical modelling + Computation and visualisation

*Special thanks to: Centre for e-Research, University of Auckland*
*New Zealand eScience Infrastructure (NeSI)*