**Using python code for analysis**

**Software requirements**

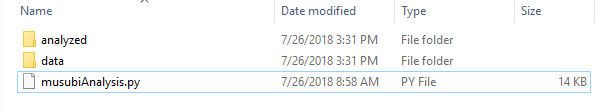
Microsoft excel, or alternative that has the ability to save spreadsheets as a .csv.

Anaconda for Python 3.6 or above. Make sure to get the python 3.6 version. Available here: <https://www.anaconda.com/download/>

Labchart. For lengths files, just the reader is okay, but to do the whole process, the full version with spike histogram is needed.

**Setting up the directory**

It should be set up like this in the drive as an example, but the directory with the code should look like this:



With a .py file from the drive, and folders “data” and “analyzed”. The “analyzed” folder should be empty if you have never run the code before- it is where output files will be generated.

If running musubiVibes.py, add an extra empty folder called “vibrations”.

**Prerequisites:**

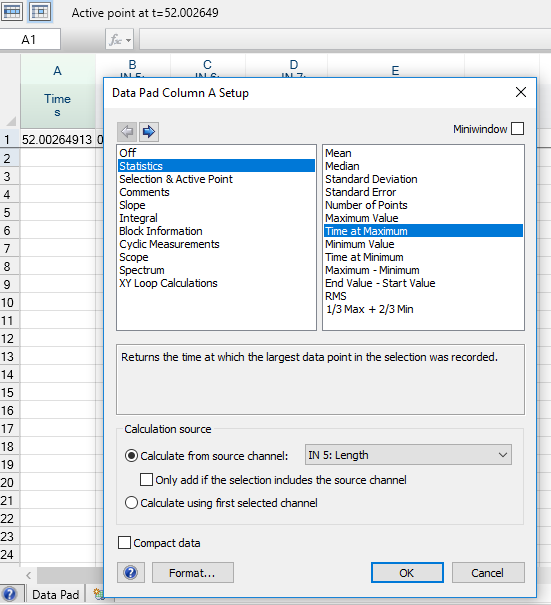
A FULLY DISCRIMINATED labchart file in the musubi or chashu macro format. The code DOES NOT discriminate action potentials or noise- it is for finding relevant statistics/measurements after a file has been discriminated.

**CSV setup**

The code requires two inputs for each file you want to analyze- one for finding markers from the lengths channel, and one containing the timestamps of the relevant action potentials.

**Lengths file**

To set up the lengths file, go into data pad view in lab chart. Delete all the data in the datapad, and then click on a column header to adjust what that column is. You should see a window like this:



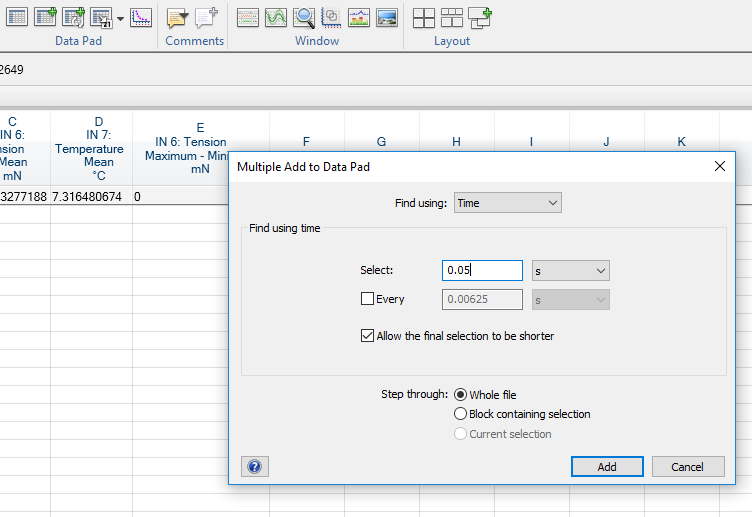
In this window, set the following columns:

For column A, under the statistics tab, select “time at maximum” from source channel “In 5: Length”

For column B, under the statistics tab, select “maximum value” from source channel “In 5: Length”

All other columns should be off (shouldn’t affect the final result, but will affect labchart’s performance and waste your time if you don’t)

Once these are set, close the column setup window. Hit “multiple add to datapad” under the “data pad” tab at the top of the labchart window. A pop up should appear that looks like this:



Set the window to exactly as it appears above. The important things to set are:

“Find using:” menu set to Time.

“Select:” set to 0.05.

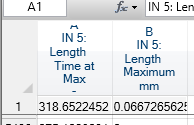
UNCHECK EVERY!!!

“Whole file” is selected.

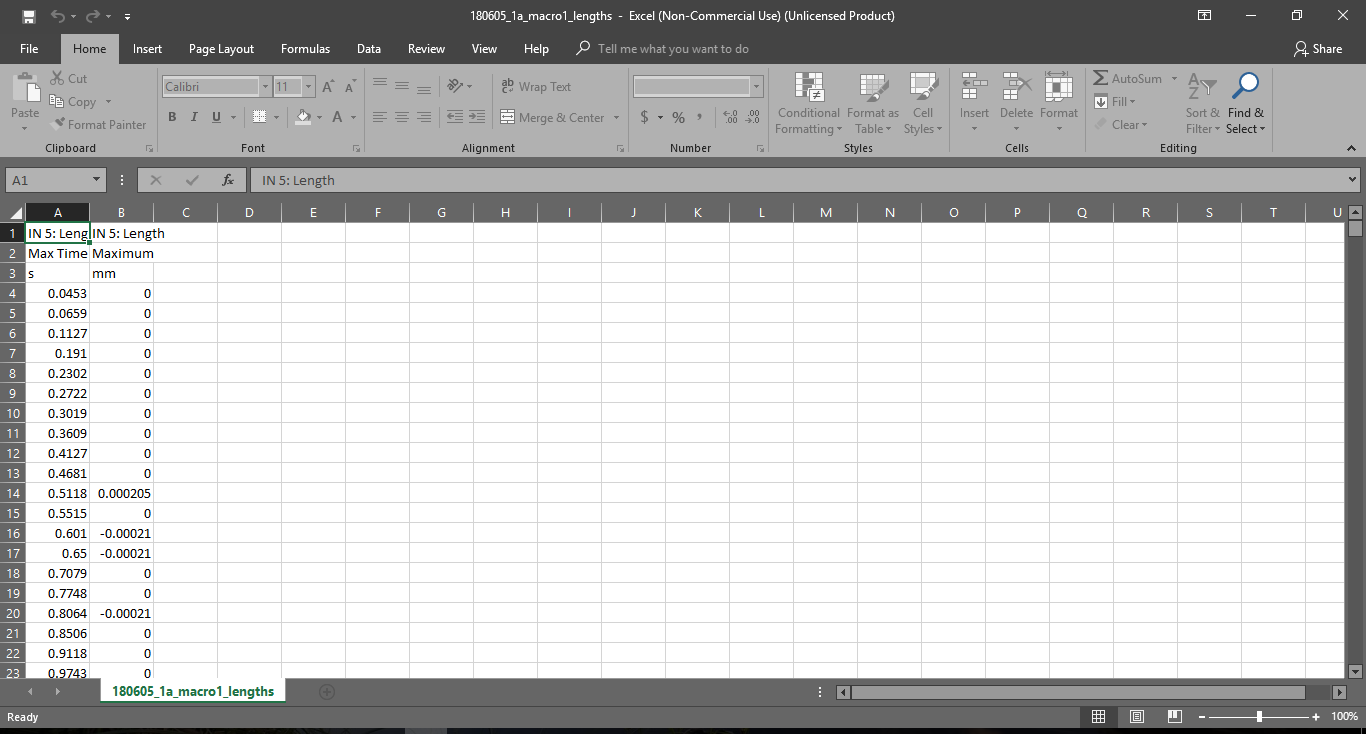
Once that is set, hit add. The datapad will fill with values.

Once the multiple add has finished, use the grey corner at the top left to select the entire datapad, and copy it into excel.

Grey corner and columns when set up:



What it looks like copied into excel:



Name the file as (labchartName)\_lengths. Save as type CSV(MS-DOS). Please, PLEASE save it ending it \_lengths- not \_length, not \_Length.

The file name should be as the end:

“(filename)\_lengths.csv”

Save it in the “data” folder that is in the same directory as the code, the one shown in the folder setup above.

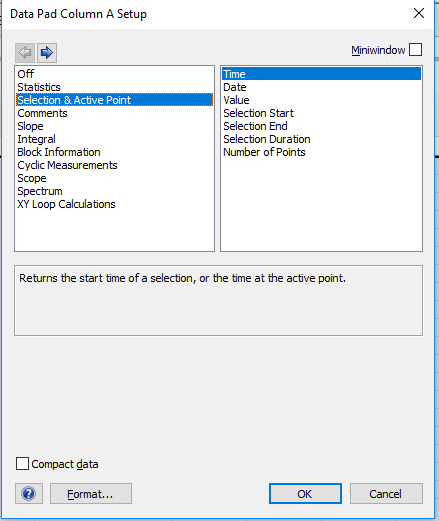
Some operating systems don’t display the .csv part. If this is the case, they will say what type of file they are in another column.

**Output file**

The steps for the output file are mostly the same, EXCEPT for what data goes in the datapad columns, the settings of multiple add to data pad, and what to name the file.

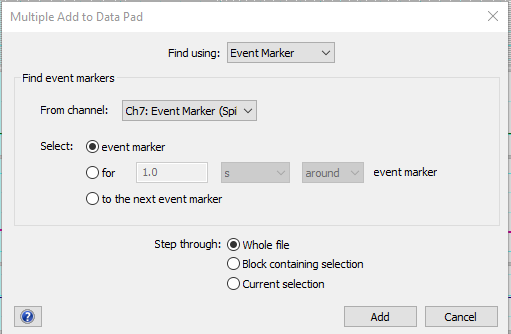
Datapad columns:

Column A should be as “Time” under the “selection and active point” tab in the column setup window. No source channel should be required. Like so:



That’s the only column to set up!

Multiple add to datapad setup:



The important things:

“Find using:” menu set to Event Marker.

“From channel:” Ch7: Event Marker (Spike Histogram)

“Whole file” is selected.

Copy it into excel in the same way.

Save as type CSV in the data folder, except the file name should be (labchart name)\_output. Not \_Output, not \_outputs, \_output. With an underscore, lowercase, and no s at the end.

**End result:**

Two files, both of type “CSV” or “Comma separated value”, with the same name up EXCEPT one should end with “\_lengths” and one should end with “\_output”.

An example fiel setup is in the drive.

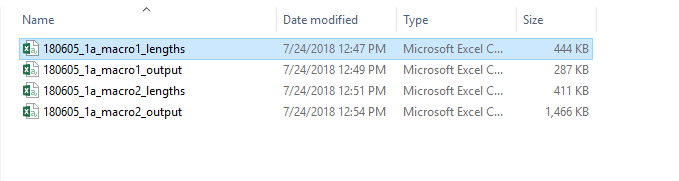
**Splitting Musubi files:**

For most musubi files, it is easier to split the labchart file into multiple chunks and analyze it that way. This is not a problem for the code- set up all the files as you would normally, just make sure of a couple of things when naming them:

-The file names should be in alphanumeric order. So for example, the first section should have a “1” in the same spot in the name as the second section would have a “2”

-The date and muscle number must always be the first two sections of the name, and “output” or “lengths” must always be the last. This is so the code knows what is what.

Example of split file setup:



Make sure, however, that there are the same number of lengths and output files as an input. Don’t split the lengths and not the outputs, or vice versa.

**Splitting Musubi CSVs without splitting the labchart file**

If you have split the CSV without splitting the labchart file, the time stamps will all be starting from the same point instead of starting over from 0, and the code will overcompensate. For this situation, set up the files the same as you would above, but add “continuous” somewhere in the file name in between the date/muscle info and the length/output info. Example setup:

x

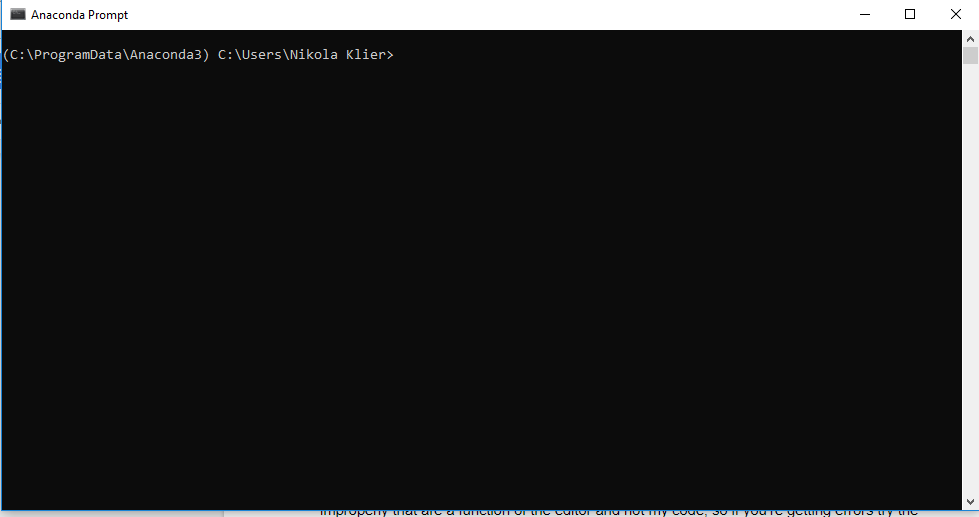
**Running the code**

Note: I prefer to use the Anaconda prompt to run my code, because sometimes running it through an editor like spyder doesn’t properly recognize certain import statements, mostly ones used for graphing. This means that some code that fails to run in an editor, will run in the anaconda prompt. If you have a better way of running python scripts that you personally like, go ahead, but if you run into issues try this.

Put as many length and output files in the data folder as you want, the code will analyze them all at once, just make sure there are the same number of length and output files and they are all paired properly. Also, make sure length and output files are the only things in the folder.

If you have an editor like Jupyter or Spyder that you know how to use and can run code from it, go ahead and use it. Just note that sometimes they have random errors from importing modules improperly that are a function of the editor and not my code, so if you’re getting errors try the DOS window. If you have no idea what this paragraph is about, you can safely ignore it.

Use the windows search tool to search for “anaconda prompt” and start the program. You should get a window like this:



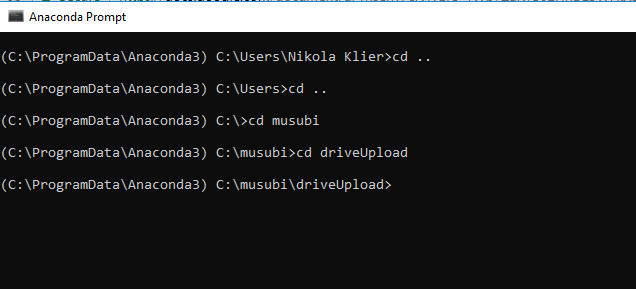
Navigate to the directory the code is in using DOS commands. The ones you need to know are:

cd ..

cd (directory name)

And hit “enter” to run commands.

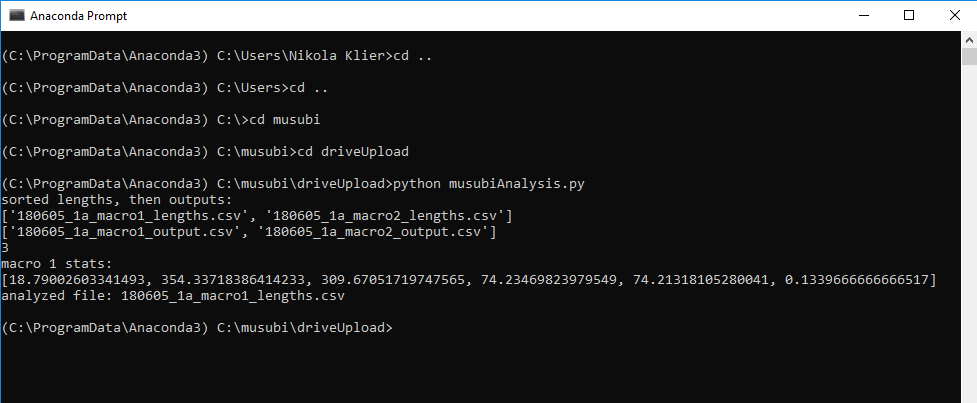
So for example, the code I want to run is in C:\musubi\driveUpload. So my window would look like:



Once in the directory, use the following command to run the code: **python (code name).py**

A bunch of text will gradually appear, telling you errors, which files have been run, and some values that I needed for debugging at some point and forgot to delete the line that causes them to show.

For example, I am running musubiAnalysis.py, so this is what it would look like:



**You’re done!!!!**

Retrieve your analyzed files from the “analyzed” folder.

**Common errors and fixes**

-Setting up the CSV files early can be really helpful if labchart is slow, crashing, or not saving your data, as it is a way to essentially save your data without saving the labchart file.

-Make sure you are using the most recent version of the code from the drive. Check the change and issues log document to see when the code or things related to the code were last analyzed.

-Check your file names. Check that you have “\_output” and “\_lengths”, and the same number of each. If you don’t, weird errors happen.

-Check your file columns: this usually causes an “index out of bounds” or “division by zero” error. ESPECIALLY check the lengths file, second column. For some reason that one gets screwed up a lot.

-CHECK YOUR FILE NAMES. Please.

-Check that nothing but output and lengths CSVs are in the data folder. If it tries to analyze something that isn’t actually data, weird things happen.

-Check your file names. Do it. Again. Please? This has happened a lot.

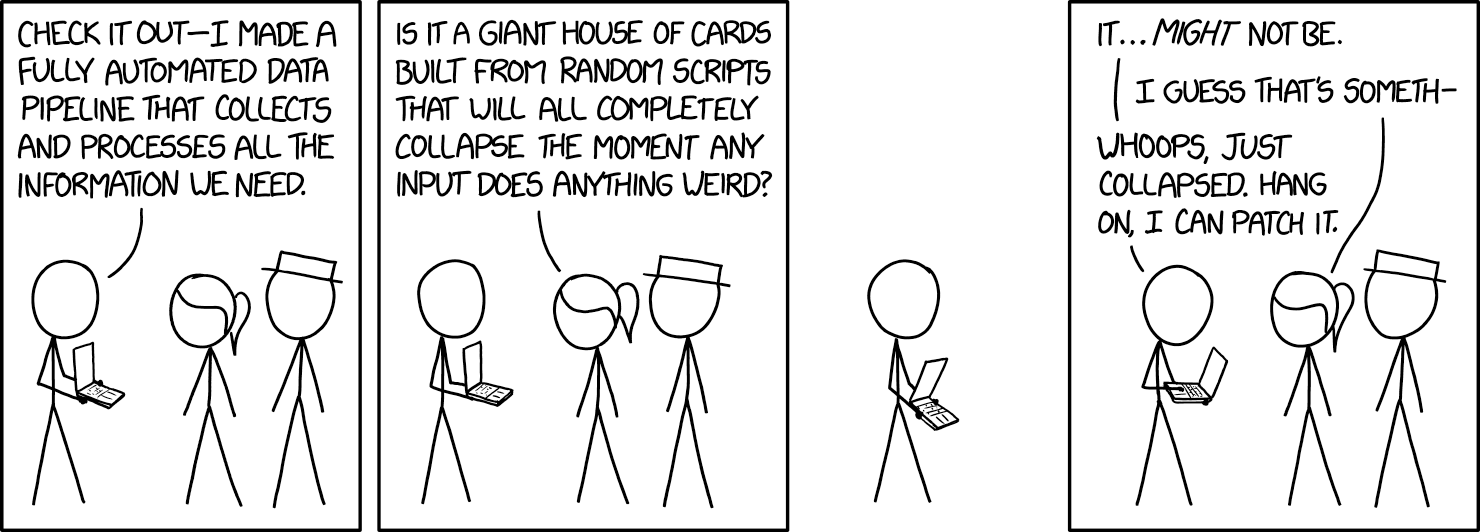
-Check you are in the right directory with the right code. It sounds stupid, and I’ve done it a stupid number of times.

-Did I mention checking your file names? Seriously, it’s how the code identifies what data it’s analyzing in the first place. It’s important.

-If none of these work, contact me (Nikola), unless I’m on vacation, in which case, take a brisk walk and find the patience to wait until I get back or analyze it by hand.

-After your walk, check the file names.

And after all of this, there will still be times where the code needs to be adjusted, so absolutely contact me if you need help. This comic is very applicable:



From <https://xkcd.com/2054/>

Happy analyzing!