# Manual linescan generation using FIJI

## Introduction

This is a protocol to explain how to manually construct demographs from individual, hand-drawn linescans. This will largely use the software FIJI [1], which is an upgraded form of ImageJ with additional plugins [2]. You can download FIJI from ImageJ's site here. The overall process for making a demograph is to generate linescans of individual cells, line them up from shortest to longest cell lengths, then display them as a heatmap of either intensity values or z-score. A visual summary of the protocol is portrayed below for clarity (**Figure 1**).

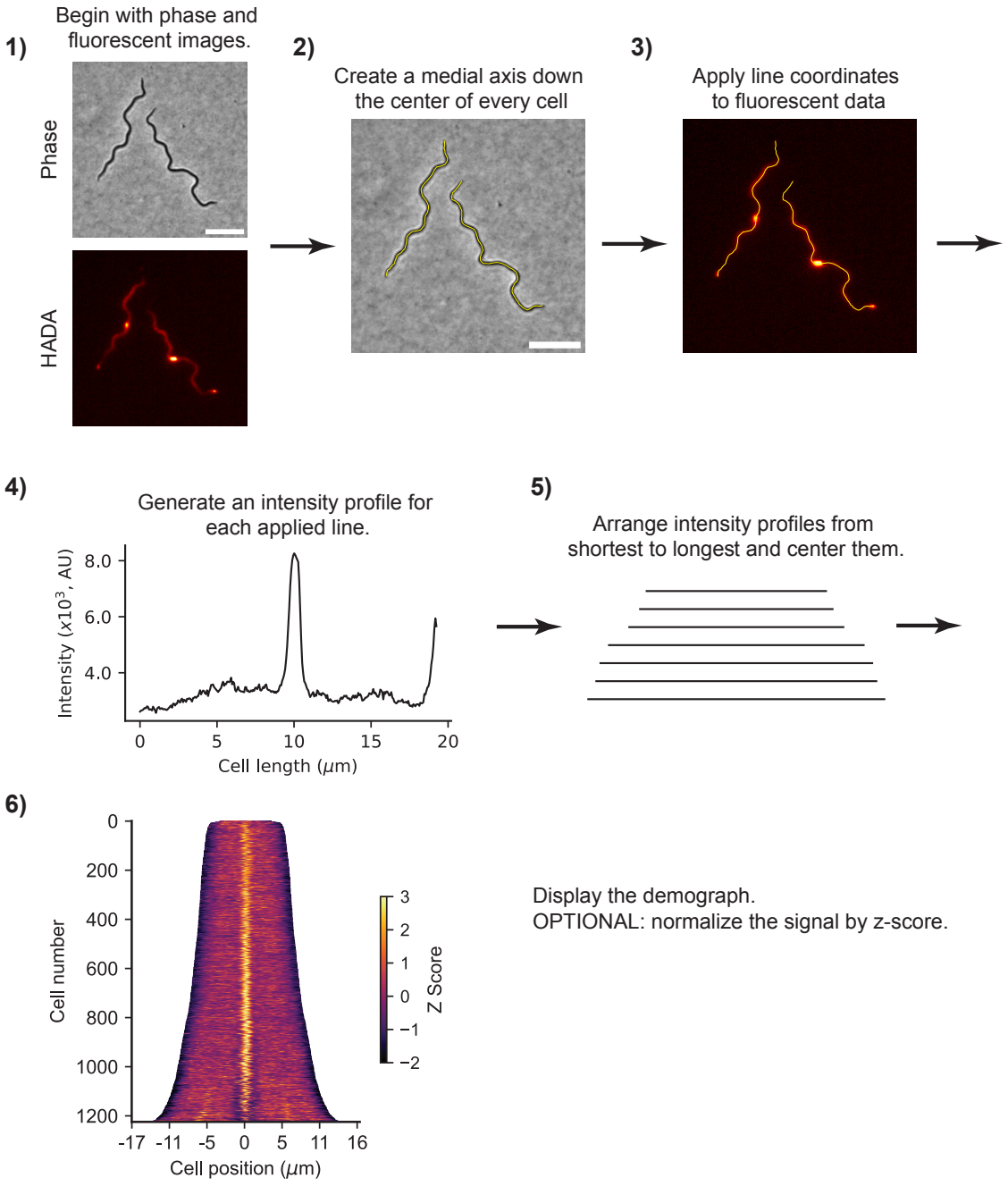



**Figure 1:** Summary workflow of demograph generation. The process begins by separating your phase and fluroescent images in **(1)**. There, trace medial axes in the phase contrast image **(2)** and apply those overlays to the fluorescent image in **(3)**. Generate intensity profiles for each linescan and save those individual files **(4)**. Arrange and center those linescans from shortest to longest **(5)**, then import them into your software of choice for plotting **(6)**. This example demograph is of a cell wall labeling dye. Results are unpublished, so please take care.

## Measuring cell intensity profiles

1. Open your phase contrast image and its respective fluorescent image. Right-click the line tool of FIJI and select the ”segemented line” option (**Fig 2**).

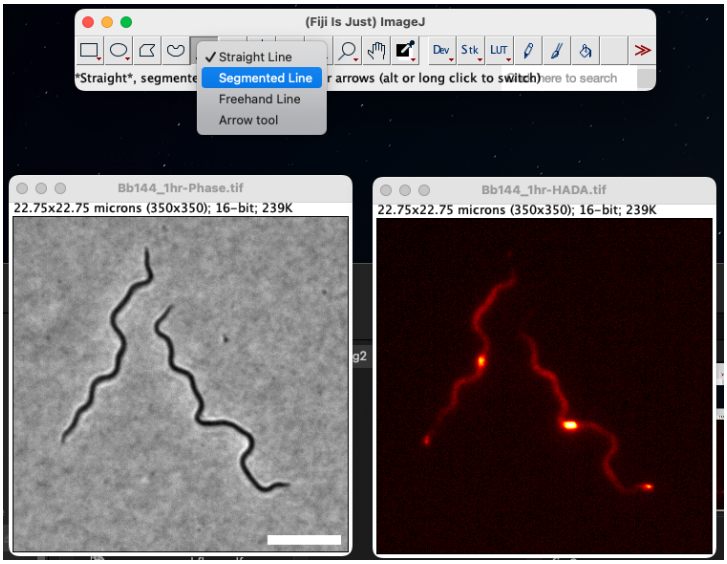



**Figure 2:** Start of the workflow. Have both the phase and fluorescent images ready and prepare the segmented line tool. Scale bar is 5$\mu$m.

2. Zoom in on your current cell and use the segmented line tool to trace the center of the cell. Double-click on the final point to end the selection process for that cell (**Fig 3**).

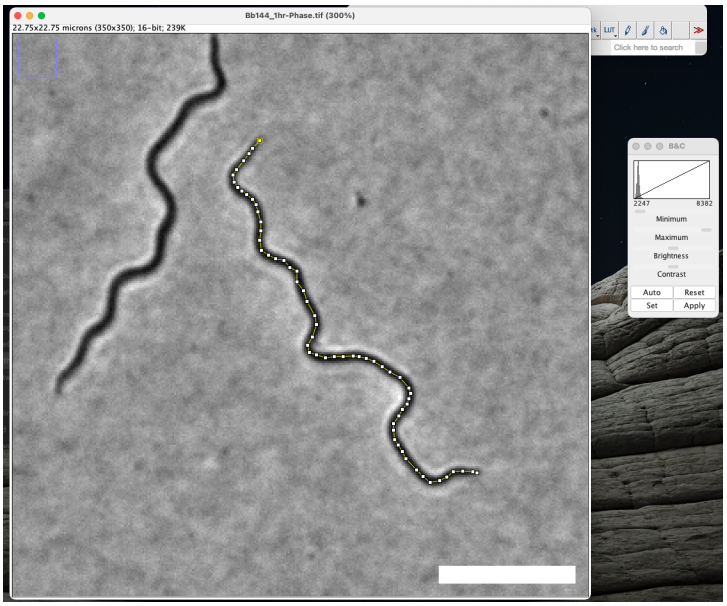
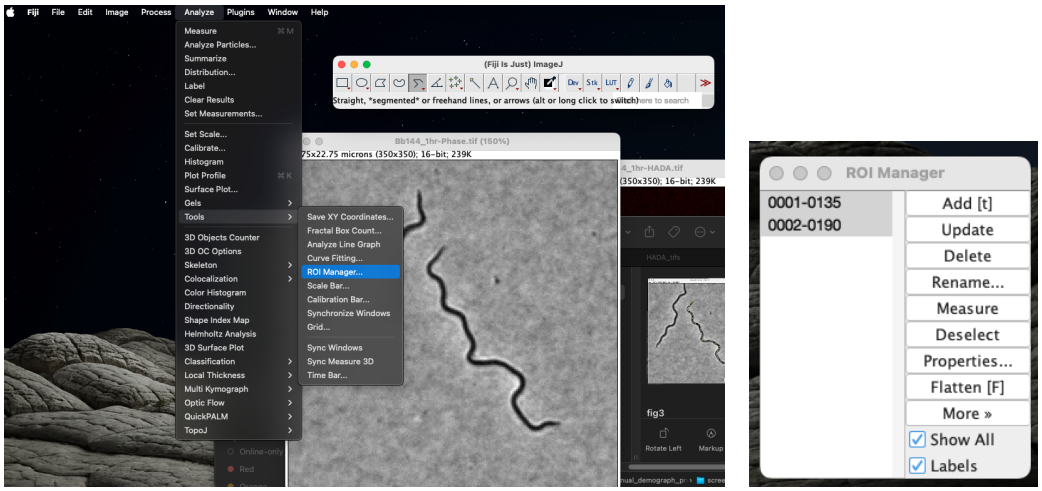



**Figure 3:** Tracing the center of the cell of interest. Scale bar is 5$\mu$m.

3. Save the generated line by adding it to your ROI list.
   - Open the ROI manage by navigating to Analyze -> Tools - ROI Manager... (**Fig 4a**).
   - In the ROI manager, click Add [t] or just press ”t” on your keyboard (**Fig 4b**).




**(a)** Where the ROI manager tool is located.

**(b)** The ROI manager.

**Figure 4:** Navigating the ROI manager.

4. Click on your fluorescent image (i.e., make it your active window). Click on the line in the ROI manager. This applies the line to the fluorescent image (**Fig 5**).

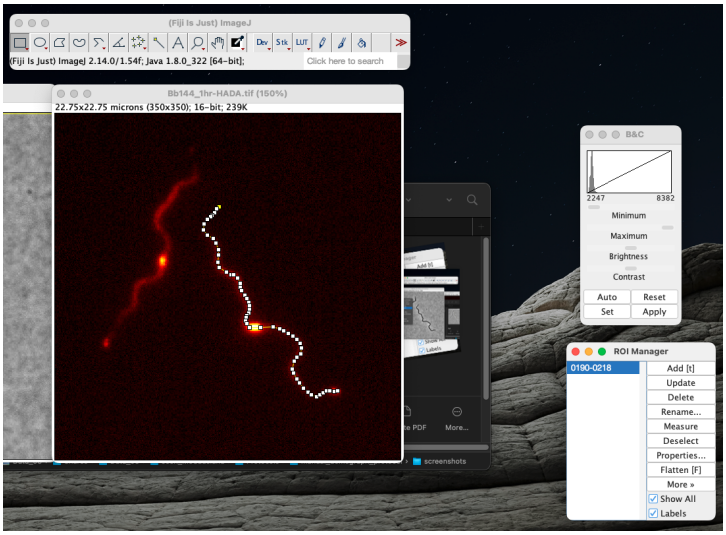



**Figure 5:** Applying the line overlay to the fluorescent data.

5. Generate an intensity profile of the fluorescent data by clicking Analyze -> Plot Profile, or just by typing ”k” on your keyboard (**Fig 6**).

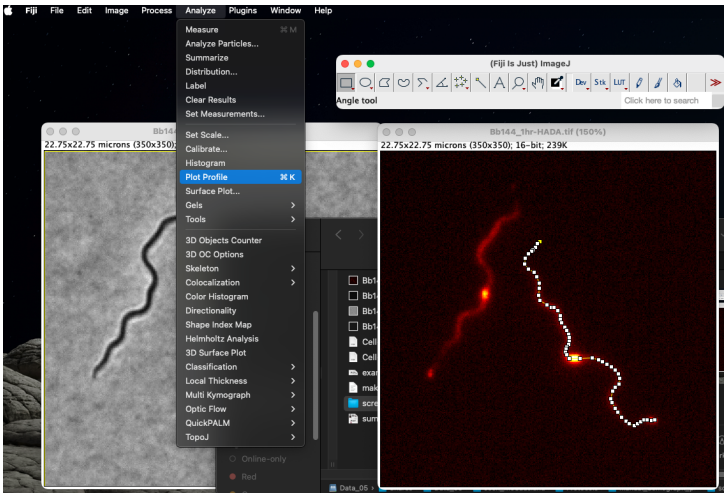



**Figure 6:** Generating the intensity profile.

6. On the generated plot, click ”Data »” -> ”Save Data...” Save this as a csv. Save this in a designated folder, ideally with a format such as ”cell-001.csv” and increment up from there. If you will measure more than 100 cells, naming them this way will easily index them in your folder.

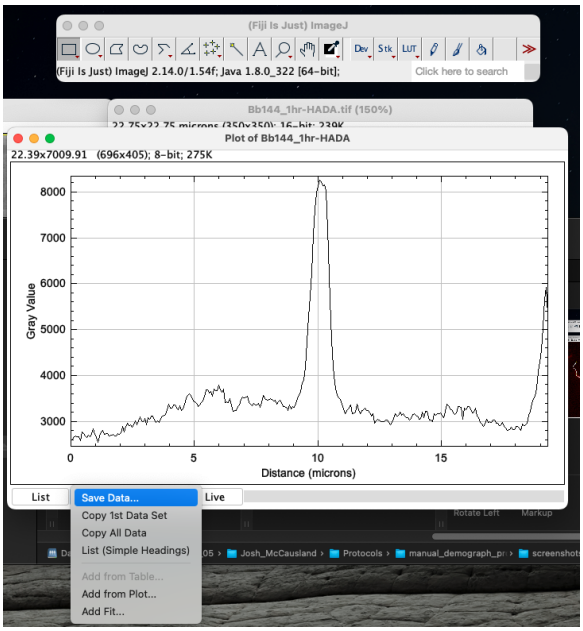



**Figure 7:** Exporting the linescan profile.

# Generating the demograph

## File organization.

Now that you have the linescans, next you must arrange them in such a way as to make the demograph. I have experimented with multiple softwares to plot demographs, including FIJI [1], but the best way to manage linescans and plot them would be through coding in R, Python, Matlab, etc. Here, I will provide a streamlined, simple code for analyzing this in python.

First, it is important to be **consistent** and **clear** in file organization and naming for the sake of coding. See the figure below for guidance (**Fig 8**).

- Have main ”folder,” or directory, where your experiment is located. The code will be deposited here too, the ”.ipynb” files.
- Put all linescans in the ”linescans” folder. The code will look for that.
- Have every linescan CSV file saved as ”Cell-xxxx.csv,” where x’s denote 0 padding. For example, linescan 1 would be ”Cell-0001.csv.” You can use a different number of 0s depending on how many manual linescans you generate. If you expect to make at least 1000 linescans, write ”Cell-0001.csv,” but between 999 and 100 linescans should be ”Cell-001.csv.”
- It is very important that the intensity values be saved under a column named ”Gray_Value,” for the code looks for this column in analysis.

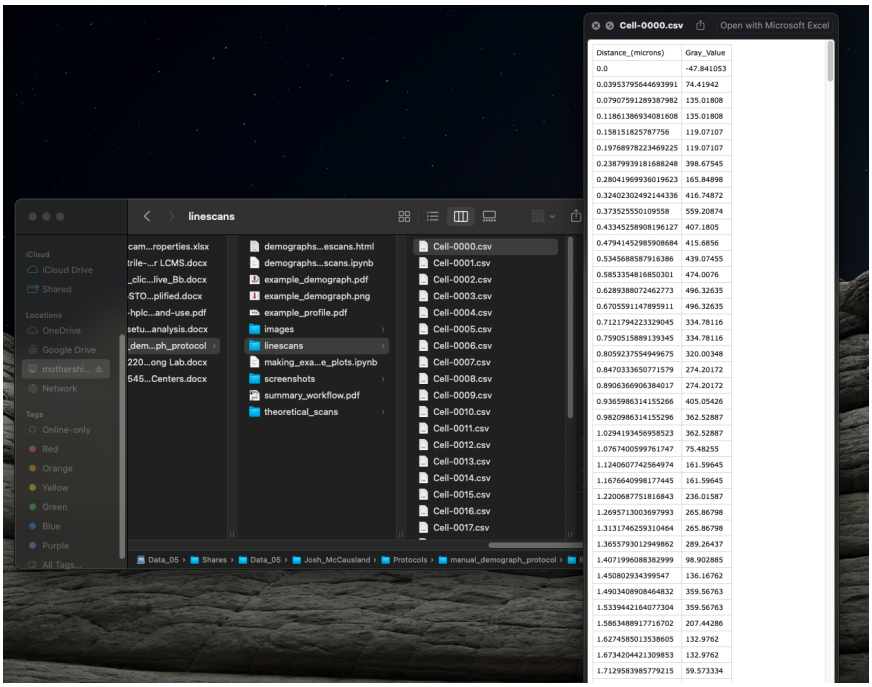



**Figure 8:** Broad organization to consider.

## Software.

Next, we will use python to analyze and plot the data. Python is free and is distributed through Anaconda, which you can download here.

- Once it is installed, open the application ”Anaconda Navigator” (**Fig 9**).
- All my code is written for Jupyter Notebook, which can be run through either the Jupyter Notebook application or VS Code.
- For a first-time user, I recommend Jupyter Notebook. It will open in your default internet browser once you start it.

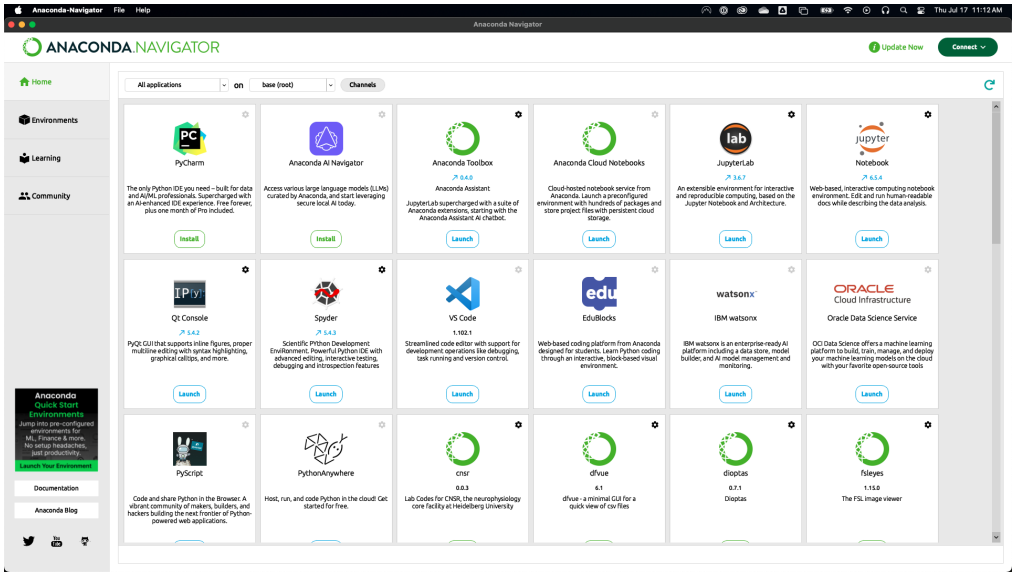



**Figure 9:** Anaconda Navigator’s home page.

Upon opening Jupyter Notebook, it will take you to your computer’s root directory. Click on folders to navigate to your experiment folder with the code, named ”demographs_from_linescans.ipynb.”

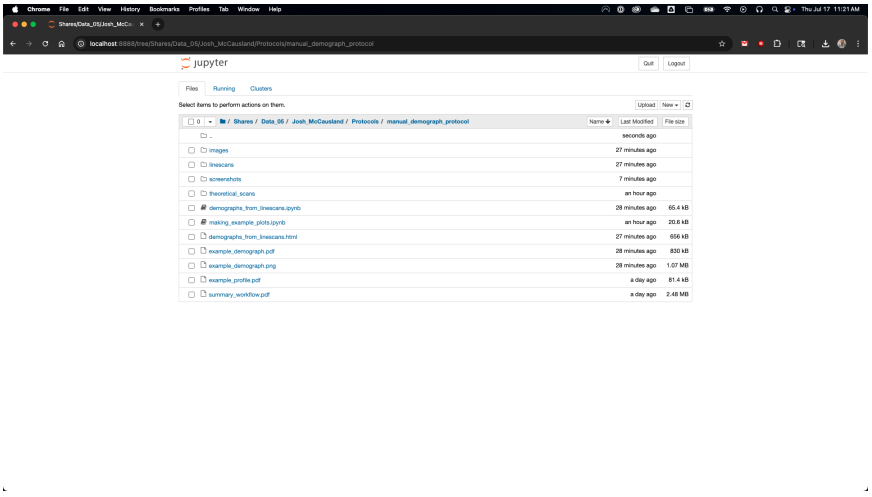



**Figure 10:** Navigate to the experiment directory by clicking on folders.

Click on the code, ”demographs_from_linescans.ipynb,” to open it.

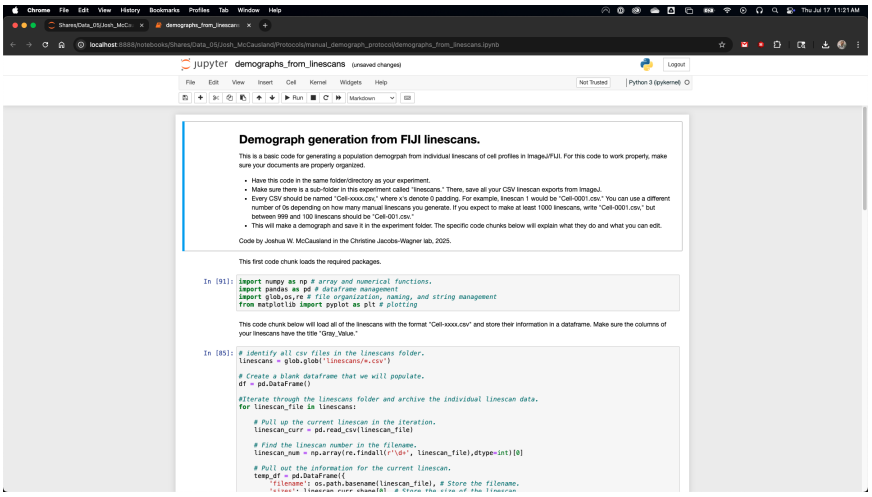



**Figure 11:** The code is now open and ready to run.

## Working with Jupyter Notebooks

Working with python, specifically Jupyter Notebooks, is quite easy. Click on a code chunk, hit ”ctrl” + ”Enter” on your keyboard, and that code chunk will run. If there is an error, Jupyter will print it for you to see. Follow the instructions written in the code, and if you need assistance let me know.

When writing methods later, it is important to cite the packages used. I used pandas to create a dataframe of the linescans [3] and organized them into a demograph using numpy [4]. Demographs were plotted with matplotlib [5].

# References


[1] **Schindelin J**, **Arganda-Carreras I**, **Frise E**, **Kaynig V**, **Longair M**, **Pietzsch T**, **Preibisch S**, **Rueden C**, **Saalfeld S**, **Schmid B**, **Tinevez JY**, **White DJ**, **Hartenstein V**, **Eliceiri K**, **Tomancak P**, and **Cardona A**, 2012. Fiji: an open-source platform for biological-image analysis. Nature Methods **9** (7):676–682.

[2] **Schneider CA**, **Rasband WS**, and **Eliceiri KW**, 2012. NIH Image to ImageJ: 25 years of image analysis. Nature Methods **9** (7):671–675.

[3] **pandas development team T**, 2020. pandas-dev/pandas: Pandas.

[4] **Harris CR**, **Millman KJ**, **van der Walt SJ**, **Gommers R**, **Virtanen P**, **Cournapeau D**, **Wieser E**, **Taylor J**, **Berg S**, **Smith NJ**, **Kern R**, **Picus M**, **Hoyer S**, **van Kerkwijk MH**, **Brett M**, **Haldane A**, **del Río JF**, **Wiebe M**, **Peterson P**, **Gérard-Marchant P**, **Sheppard K**, **Reddy T**, **Weckesser W**, **Abbasi H**, **Gohlke C**, and **Oliphant TE**, 2020. Array programming with NumPy. Nature **585** (7825):357–362.

[5] **Hunter JD**, 2007. Matplotlib: A 2D graphics environment. Computing in Science & Engineering **9** (3):90–95.