

README for bin_size_optimization.py and categorization_algorithm.py

Outline:

- 1) Install Python 3
- 2) Download the script files
- 3) Format and save data set
- 4) Run bin_size_optimization.py and interpret results
- 5) Run categorization_algorithm.py and interpret results
- 6) Explanation of main functions

1) Install Python 3

- a. Since these scripts make use of several Python 3 libraries (tkinter, csv, random, os, sys, numpy, and math), the simplest way to install Python 3 with these required libraries is through the Anaconda distribution of Python 3 which can be found here (<https://www.anaconda.com/download/>).
- b. Download the appropriate installer (.exe) for your operating system (32-bit or 64-bit - Windows or Mac)
- c. Once the installer (.exe) has finished downloading, open it and follow the click-through instructions to install the Anaconda distribution of Python 3. An example of the first step of the installer for a 64-bit Windows system is shown below.

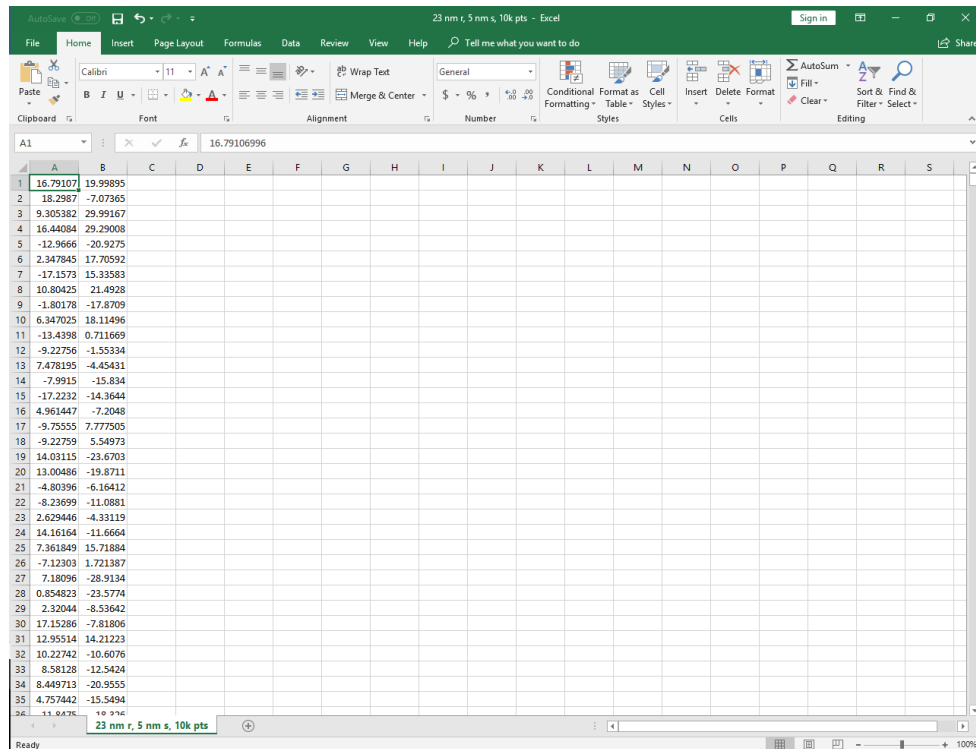


2) Download the script files

- a. Download the script files “bin_size_optimization.py” and “categorization_algorithm.py” and save them into a folder on your computer.

3) Format and save data set

- a. The scripts only accept .csv files formatted with the x values in the first column and y values in the second column. The y axis should be the axis oriented in the same direction as the r axis. An example of the .csv file is shown below. An experimental data set for Importin B1 and a simulated data set (23 nm mean, 5 nm s.d., 10,000 localizations) are provided also for aid.

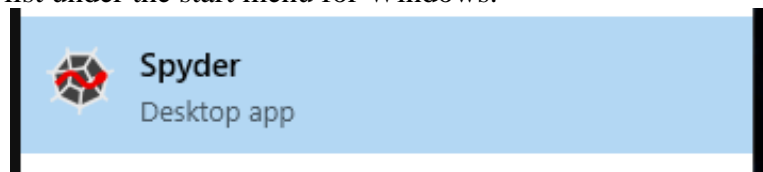


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	16.79107	19.99895																	
2	18.2987	-7.07365																	
3	9.305382	29.99167																	
4	16.44084	29.29008																	
5	-12.9666	-20.9275																	
6	2.347845	17.70592																	
7	-17.1573	15.33583																	
8	10.80425	21.4928																	
9	-1.80178	-17.8709																	
10	6.347025	18.11496																	
11	-13.4398	0.711669																	
12	-9.22756	-1.55334																	
13	7.478195	-4.45431																	
14	-7.9915	-15.834																	
15	-17.2232	-14.3644																	
16	4.961447	-7.2048																	
17	-9.75555	7.77505																	
18	-9.22759	5.54973																	
19	14.03115	-23.6703																	
20	13.00486	-19.8711																	
21	-4.80396	-6.16412																	
22	-8.23699	-11.0881																	
23	2.629446	-4.33119																	
24	14.16164	-11.6664																	
25	7.361849	15.71884																	
26	-7.12303	1.721387																	
27	7.18096	-28.9134																	
28	0.854823	-23.5774																	
29	2.32044	-8.53642																	
30	17.15286	-7.81806																	
31	12.95514	14.21223																	
32	10.22742	-10.6076																	
33	8.58128	-12.5424																	
34	8.449713	-20.9555																	
35	4.757442	-15.5494																	

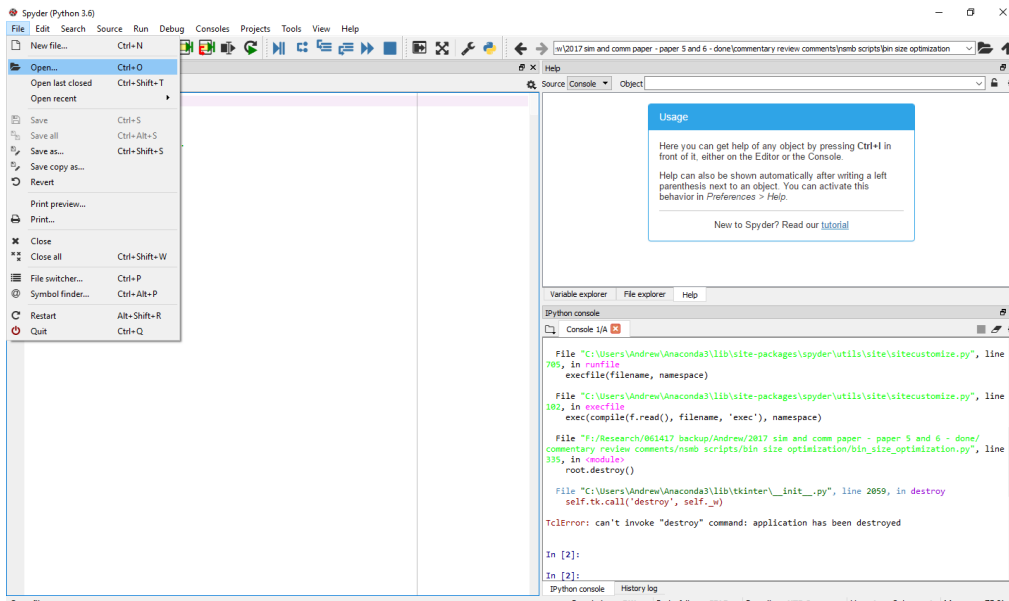
- b. Save the .csv file with the xy data to your computer.

4) Run bin_size_optimization.py and interpret results

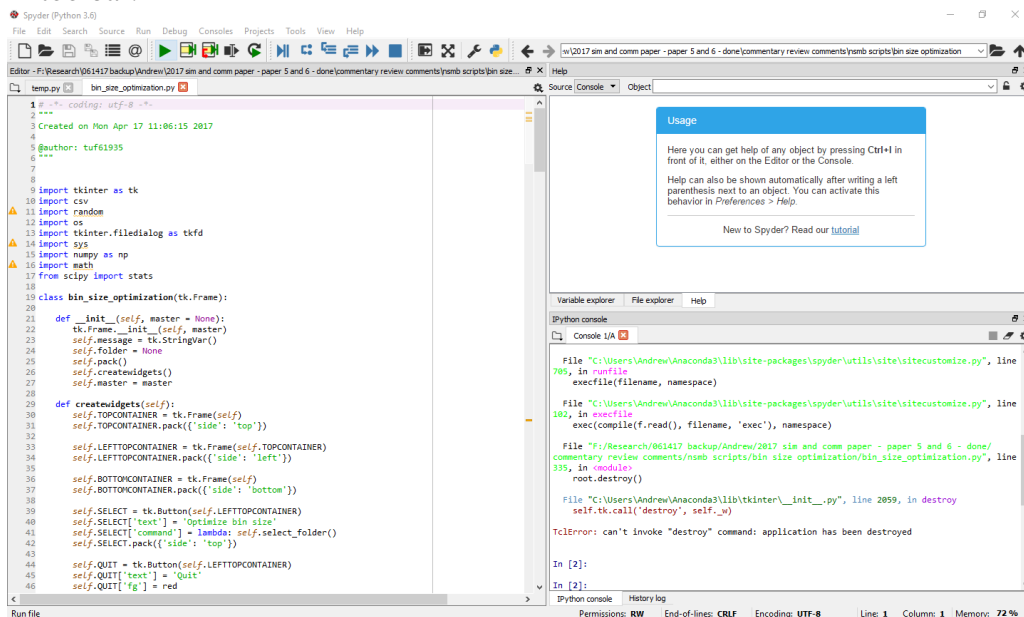
- a. Packaged in the Anaconda distribution of Python 3 is an integrated development environment for Python 3 called “Spyder” which was installed on your computer. Open Spyder now – it should have an icon on your desktop or an icon in the program list under the start menu for Windows.



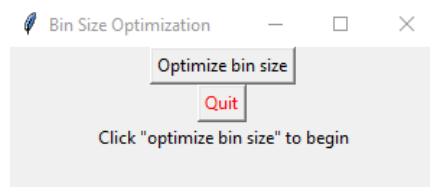
- b. Once Spyder starts up, open “bin_size_optimization.py” using the File>Open menu.



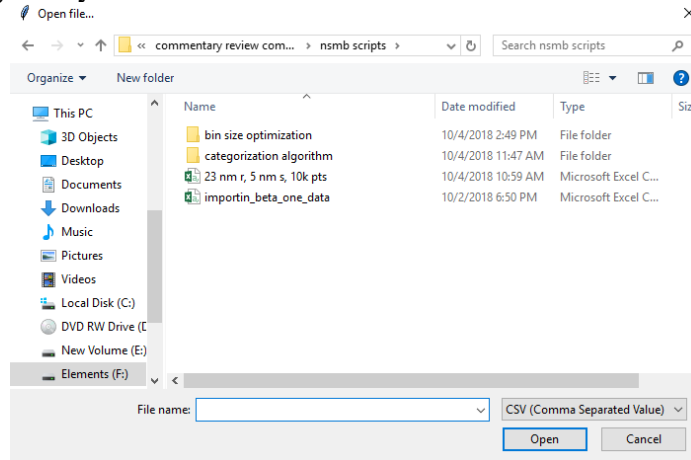
c. Once “bin_size_optimization.py” is open, click the green “Run file” arrow in the toolbar.



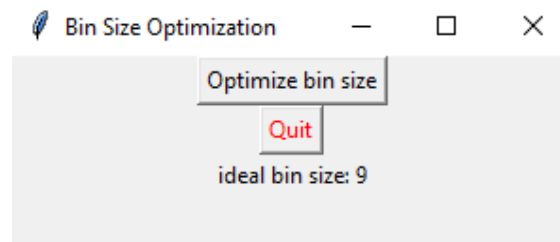
d. After the script runs, a graphical user interface (gui) should open. It may open behind Spyder, in which case either minimize Spyder or select the gui from the taskbar.



- e. Click “Optimize bin size” and using the window to navigate to the .csv file containing your xy data.

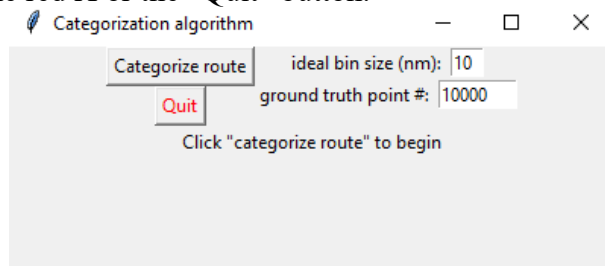


- f. After you select the .csv file with your xy data, click “Open”. After a few moments, the optimized bin size will be displayed in the gui window and another .csv file named “binsize_optimization_results.csv” with the raw data for the bin size optimization calculation will be written into the same folder alongside the “bin_size_optimization.py” script. The data in “binsize_optimization_results.csv” can be used to make the graph shown in Figure 1B.



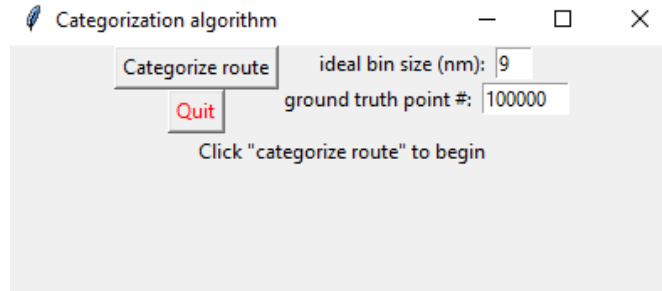
5) Run categorization_algorithm.py and interpret results

- a. Use the same steps as above to open the gui for the “categorization_algorithm.py” script. If the previous gui is already open, it is necessary to close the open gui with either the red X or the “Quit” button.

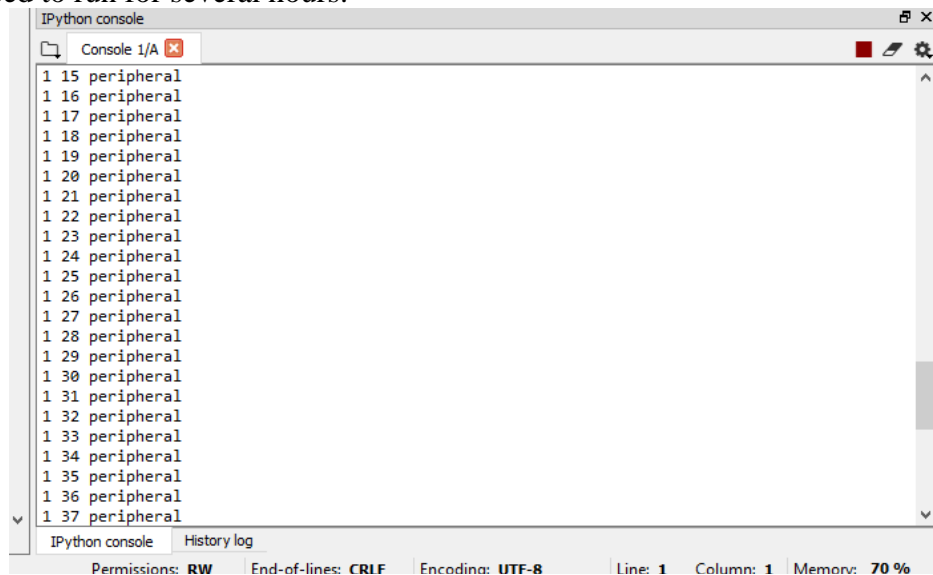


- b. Change the ideal bin size default value to the optimal bin size which was determined in the previous section. Also, adjust the number of localizations that will be simulated to calculate the ground truth distributions. 1,000 localizations may be used for a very quick calculation (several minutes) of the ground truth

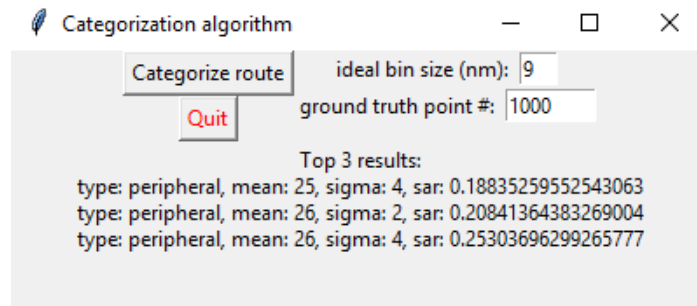
distributions but greater than 100,000 localizations should be used for a statistically accurate calculation of the ground truth distributions. Currently, the peak radius and localization precision for the ground truth distributions are simulated at integer (whole number) intervals. Ultimately, this interval reduces computational complexity while still maintaining biological relevance.



- c. Click “Categorize route”, navigate to the location of your .csv file with your xy data, and click “Open”.
- d. At this point the simulation will begin according to the parameters entered in the gui. No other values need to be provided. The progress can be monitored in the IPython console in Spyder. As mentioned above, if ~1,000 points are used to simulate the ground truth distribution, the simulation will take several minutes to complete. On the other hand, if >100,000 points are used, the simulation will need to run for several hours.



- e. After the simulation has finished, the algorithm will compare the 3D density histogram for your xy data to each ground truth 3D density histogram via sum of the absolute-valued residuals (SAR). It will then display the top 3 results in the gui and write all the results into a separate .csv file called “categorization_results.csv” sorted by SAR. This .csv file will be in the same folder as “categorization_algorithm.py”. The data from “categorization_results.csv” may be used to replicate the graph shown in Supplementary Figure 3.



6) Explanation of main functions

- a. `bin_size_optimization.py`
 - i. `gen_matrix`: generates the area matrix for calculating the 3D density histogram
 - ii. `deconvolution`: calculates the 3D density histogram from the y-dimensional histogram and the area matrix from `gen_matrix`
 - iii. `simulation`: returns the smallest bin size (starting at 1) where significant negative/error values do not appear in the 3D density histogram and writes results to a .csv.
- b. `categorization_algorithm.py`
 - i. `gen_matrix`: same as above
 - ii. `deconvolution`: same as above
 - iii. `gauss_dist`: simulates either a peripheral or bimodal distribution depending on the `dist_type` variable with a mean and standard deviation depending on the radius and precision variables
 - iv. `uniform_dist`: simulates a uniform distribution with a mean and standard deviation depending on the radius and precision variables
 - v. `categorization_algorithm`: calculates the 3D density histogram from the given .csv xy data, simulates peripheral/central/uniform/bimodal ground truth distributions across all solution space, compares the 3D density histogram from the .csv xy data to each ground truth distribution, sorts the results (smallest to largest) by SAR, and writes results to a .csv.