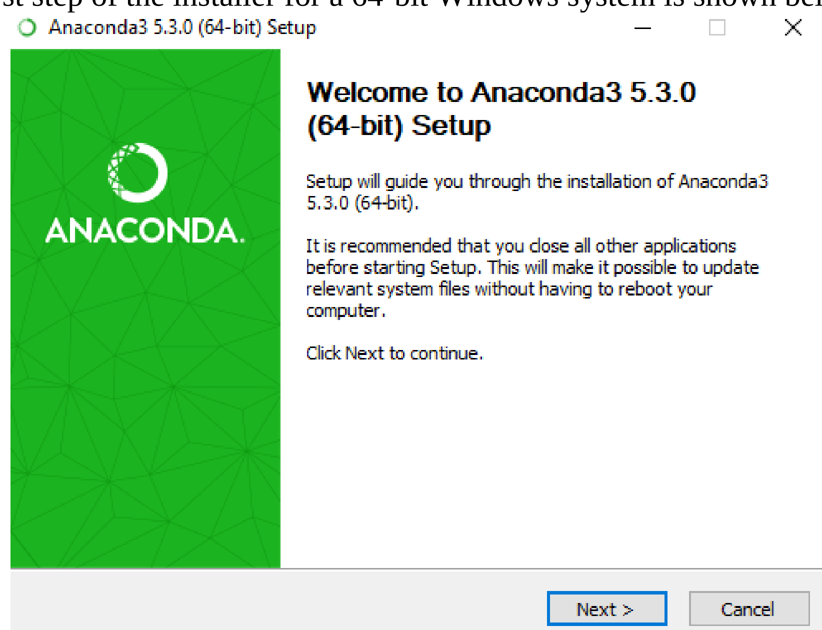
README

Outline:

1) Install Python 3
2) Download the script files
3) Figure 4 – precision simulation
4) Figure 5 – point number simulation
5) Figure 6 – route categorization – how to format and save data set
6) Figure 6 – route categorization – bin size optimization simulation
7) Figure 6 – route categorization – categorization simulation
8) Figure 7 – symmetry compression simulation
9) Figure 8 – labeling efficiency simulation
10) 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, scipy, 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.

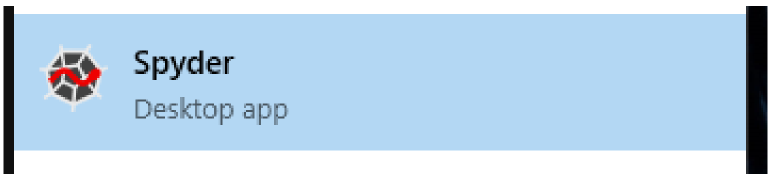


2) Download the script files

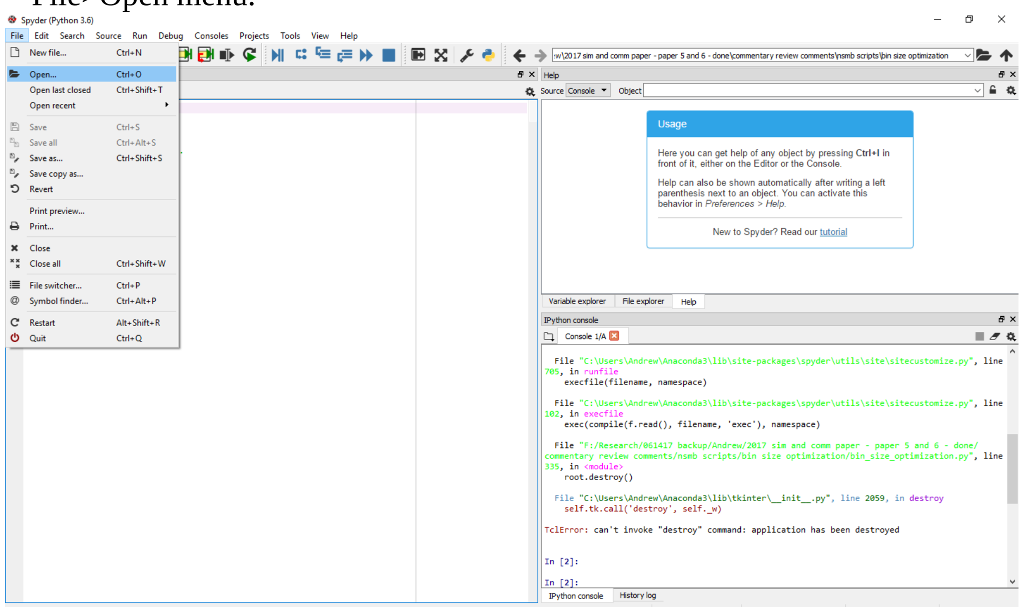
a. Download the folder containing the script files from https://github.com/andrewruba/YangLab and save it on your computer.

3) Figure 4 – precision simulation

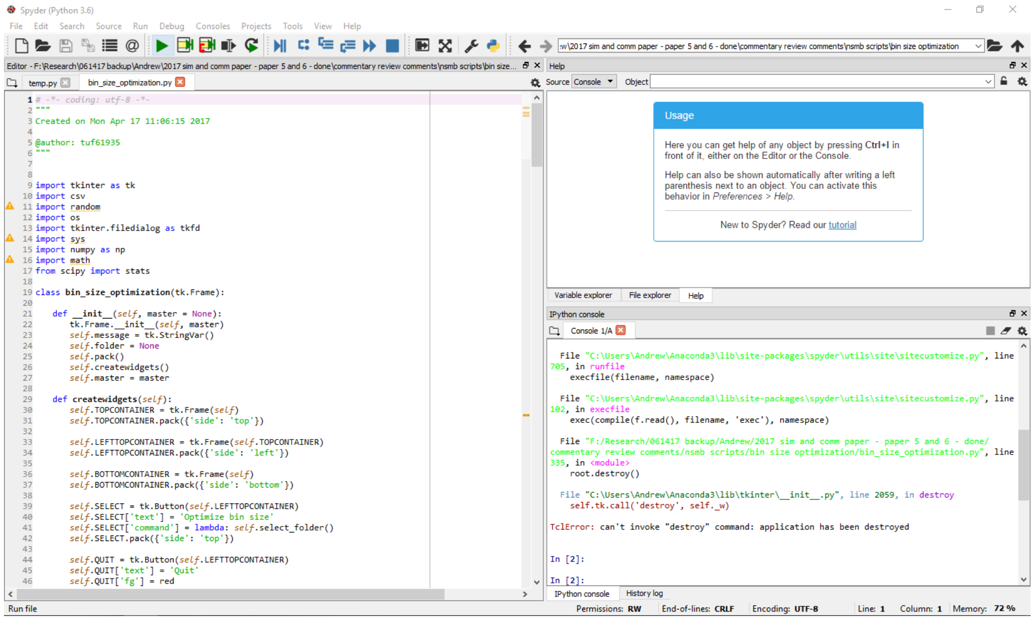
a. Packaged in the Anaconda distribution of Python 3 is an integrated development environment for Python 3 called “Spyder” which was installed on your computer. Open Spyder now – it should have an icon on your desktop or an icon in the program list under the start menu for Windows.



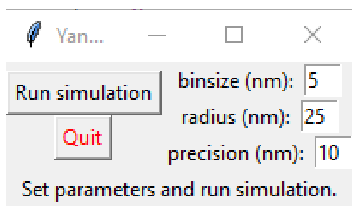
b. Once Spyder starts up, open “Figure 4 – precision\simulation_gui.py” using the File>Open menu.



c. Once “simulation_gui.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.

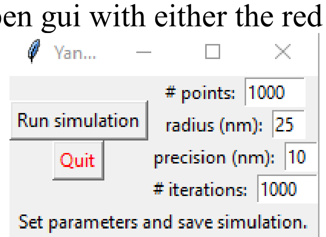


e. Click “Run simulation” after entering the proper integer values for the simulation parameters you would like to run. The bin size may be set small since 1,000,000 points are simulated.

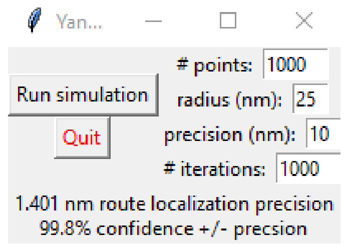
f. After a few moments, the results will be written to a .csv file named “precision_results.csv” into the same folder alongside the “Figure 4 – precision\simulation_gui.py” script. The data in “precision_results.csv” can be used to make the graph shown in Figure 4 by entering the values into any graphing program and measuring the error between a single Gaussian fitting vs a bimodal Gaussian fitting.

4) Figure 5 – point number simulation

a. Use the same steps as above to open the gui for the “Figure 5 – number of points\simulation_gui.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

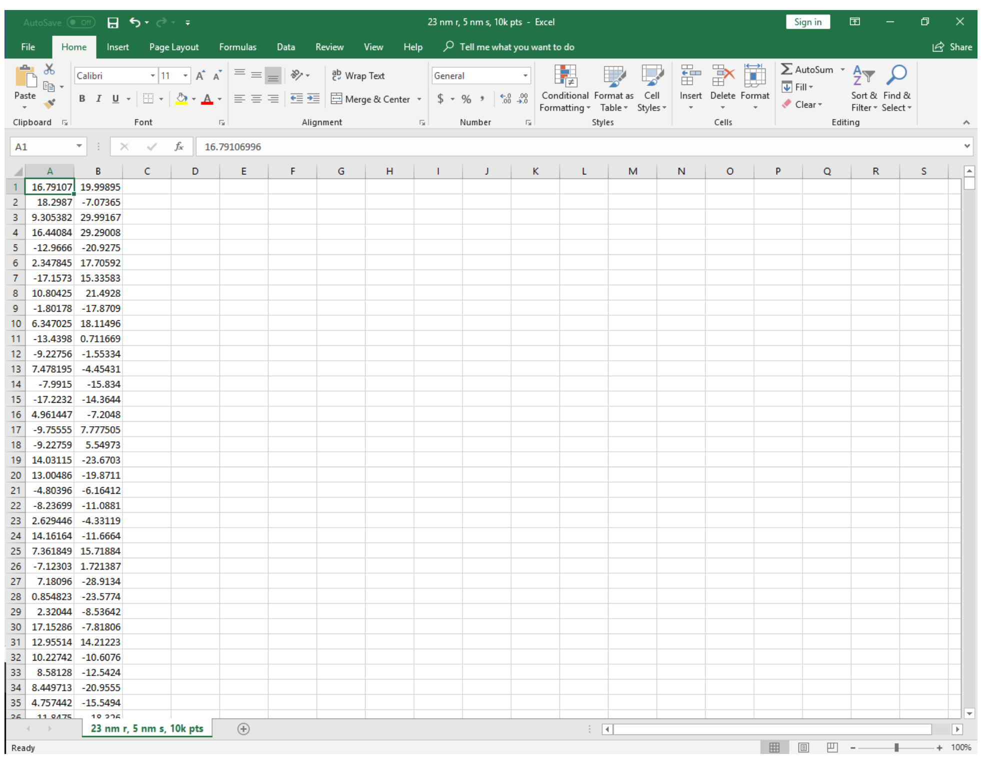


b. Click “Run simulation” after entering the proper integer values for the simulation parameters you would like to run. The optimal bin size will be dynamically calculated according to the parameters and the reproducibility rate and route localization precision will be written into the gui message area.



5) Figure 6 – route categorization – how to format and save data set

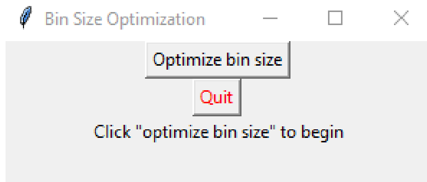
a. This script accepts .csv files formatted with the x values in the first column and y values in the second column. The y axis should be the axis oriented in the same direction as the r axis. An example of the .csv file is shown below. An experimental data set for Importin B1 and a simulated data set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



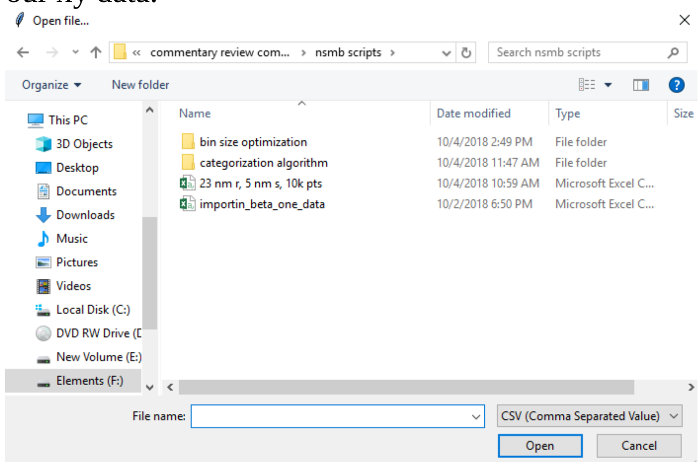
b. Save the .csv file with the xy data to your computer.

6) Figure 6 – route categorization – bin size optimization simulation

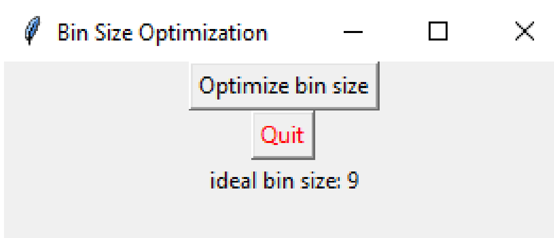
a. In accordance with another publication, the bin size optimization step is separated from the categorization algorithm. Use the same steps as above to open the gui for the “Figure 6 – route cateogorization\bin size optimization\bin_size_optimization.py” script. After the script runs, a graphical user interface (gui) should open.



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

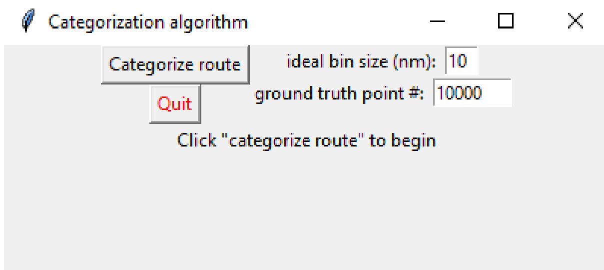


c. After you select the .csv file with your xy data, click “Open”. After a few moments, the optimized bin size will be displayed in the gui window and another .csv file named “binsize_optimization_results.csv” with the raw data for the bin size optimization calculation will be written into the same folder alongside the “bin_size_optimization.py” script. The data in “binsize_optimization_results.csv”.

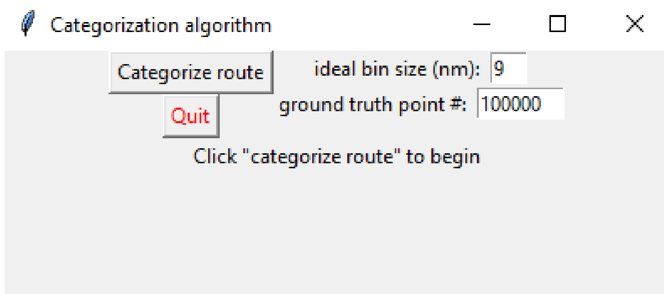


7) Figure 6 – route categorization – categorization simulation

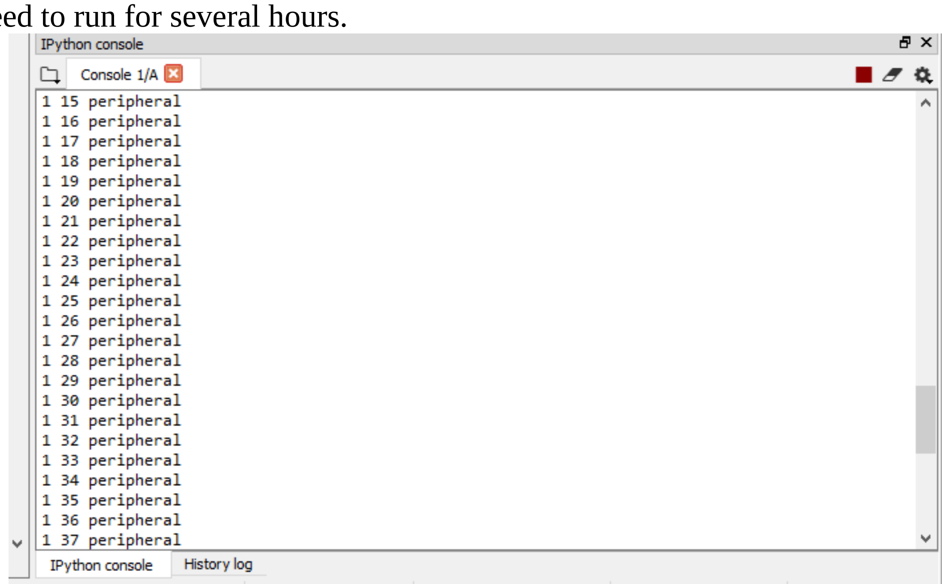
a. Use the same steps as above to open the gui for the “Figure 6 - route categorization\categorization algorithm\categorization_algorithm.py” script.



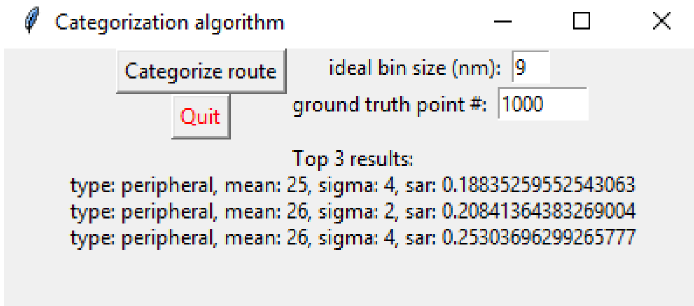
b. Change the ideal bin size default value to the optimal bin size which was determined in the previous section. Also, adjust the number of localizations that will be simulated to calculate the ground truth distributions. 1,000 localizations may be used for a very quick calculation (several minutes) of the ground truth distributions but greater than 100,000 localizations should be used for a statistically accurate calculation of the ground truth distributions.



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.

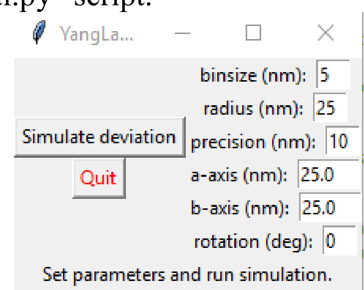


e. After the simulation has finished, the algorithm will compare the 3D density histogram for your xy data to each ground truth 3D density histogram via sum of the absolute-valued residuals (SAR). It will then display the top 3 results in the gui and write all the results into a separate .csv file called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

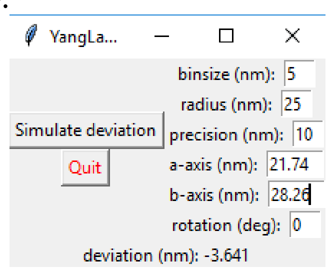


8) Figure 7 – symmetry compression

a. Use the same steps as above to open the gui for the “Figure 7 - symmetry compression\simulation_gui.py” script.

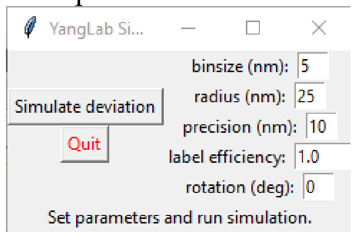


b. Click “Simulate deviation” after entering the proper integer values for the simulation parameters you would like to run. Since the simulation uses 1,000,000 points, the bin size may be set small. To calculate the values for the a and b axes, solve the system of equations where a/b = your desired ratio and the circumference of the undistorted symmetry = Ramanujan’s estimation for the a and b axes of an ellipse with equal circumference: $\pi\left[3(a+b)-\sqrt{(3a+b)(a+3b)}\right]$. In order to match the simulation in the manuscript, the a axis should be set as the smaller axis. The rotation should be set as an integer value between 0 and 360. The results will be written to the gui message.

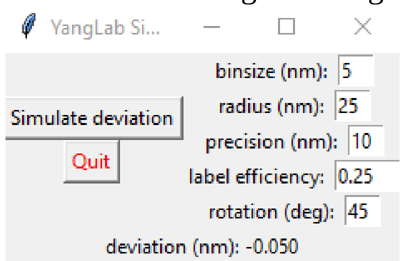


10) Figure 8 – labeling efficiency simulation

a. Use the same steps as above to open the gui for the "Figure 8 - labeling efficiency\simulation_gui.py" script.



b. Click "Simulate deviation" after entering the proper integer values for the simulation parameters you would like to run. Since the simulation uses 1,000,000 points, the bin size may be set small. The labeling efficiency should be set as a value between 0.0 and 1.0. The rotation should be set as an integer value between 0 and 360. The results will be written to the gui message.



11) Explanation of main functions

c. 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.

d. categorization_algorithm.py
   iv. gen_matrix: same as above
   v. deconvolution: same as above
   vi. 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
   vii. uniform_dist: simulates a uniform distribution with a mean and standard deviation depending on the radius and precision variables
   viii. 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.