README

Outline:

1) Install Python 3
2) Download the script files
3) Supplemental Figure 8 – precision simulation
4) Supplemental Figure 9 – point number simulation
5) Supplemental Figure 10 – labeling efficiency simulation
6) Supplemental Figure 11 – symmetry compression simulation
7) Explanation of main functions

1) Install Python 3

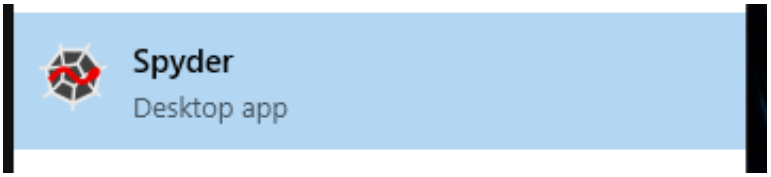
a. Since these scripts make use of several Python 3 libraries (tkinter, csv, random, os, sys, numpy, scipy, and math), the simplest way to install Python 3 with these required libraries is through the Anaconda distribution of Python 3 which can be found here (https://www.anaconda.com/download/).
b. Download 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

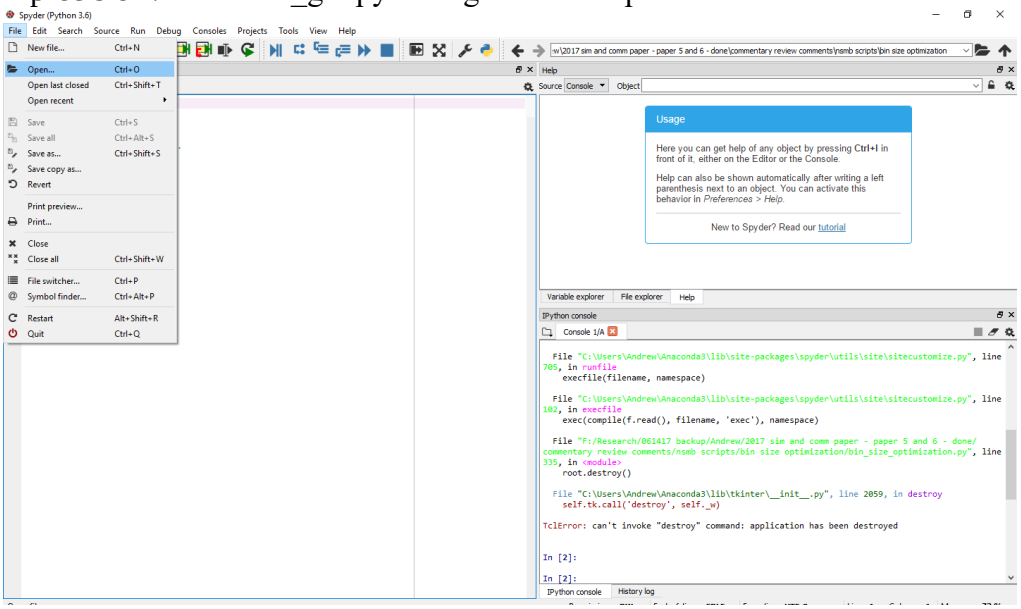
a. Download the folder containing the script files from https://github.com/andrewruba/YangLab and save it on your computer.

3) Supplemental Figure 8 – precision simulation

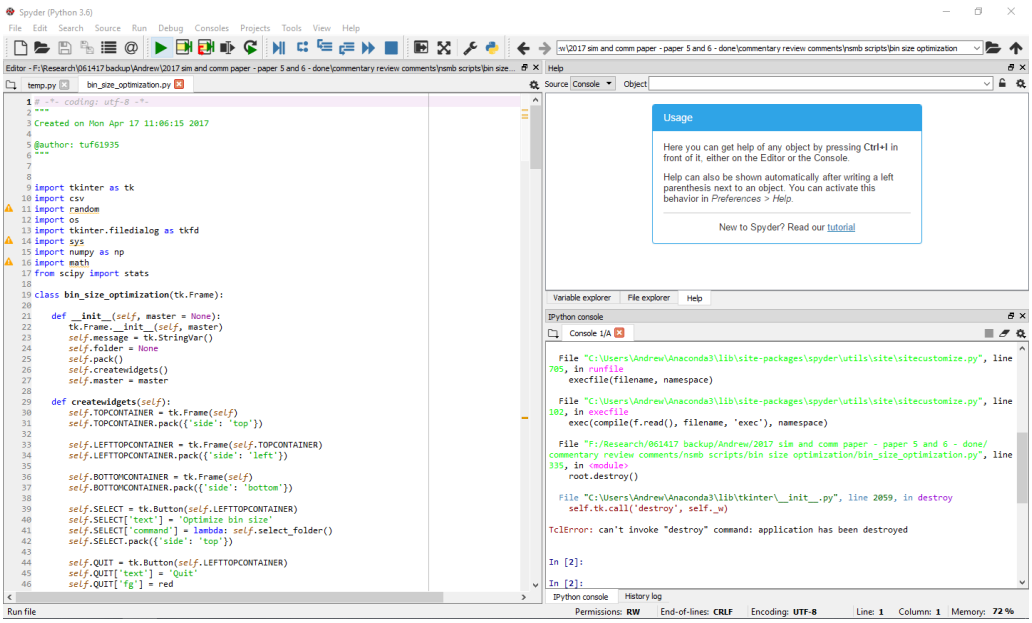
a. Packaged in the Anaconda distribution of Python 3 is an integrated development environment for Python 3 called “Spyder” which was installed on your computer. Open Spyder now – it should have an icon on your desktop or an icon in the program list under the start menu for Windows.



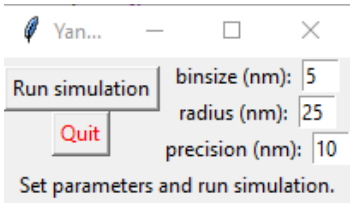
b. Once Spyder starts up, open “Supplemental Figure 8 – precision\simulation_gui.py” using the File>Open menu.



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.

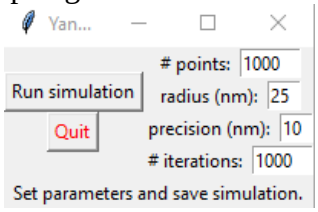


e. Click “Run simulation” after entering the proper integer values for the simulation parameters you would like to run. The bin size may be set small since 1,000,000 points are simulated.

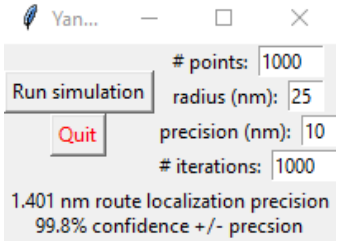
f. After a few moments, the results will be written to a .csv file named “precision_results.csv” into the same folder alongside the “Supplemental Figure 8 – precision\simulation_gui.py” script. The data in “precision_results.csv” can be used to make the graph shown in Supplemental Figure 8 by entering the values into any graphing program and measuring the error between a single Gaussian fitting vs a bimodal Gaussian fitting.

4) Supplemental Figure 9 – point number simulation

a. Use the same steps as above to open the gui for the “Supplemental Figure 9 – 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

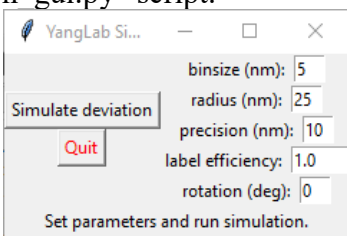


b. Click “Run simulation” after entering the proper integer values for the simulation parameters you would like to run. The optimal bin size will be dynamically calculated according to the parameters and the reproducibility rate and route localization precision will be written into the gui message area.

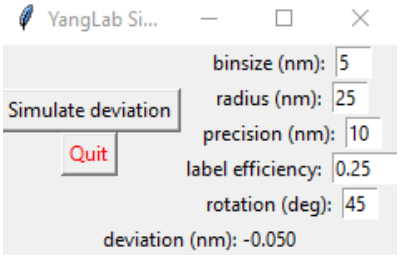


5) Supplemental Figure 10 – labeling efficiency simulation

a. Use the same steps as above to open the gui for the “Supplemental Figure 10 - labeling efficiency\simulation_gui.py” script.

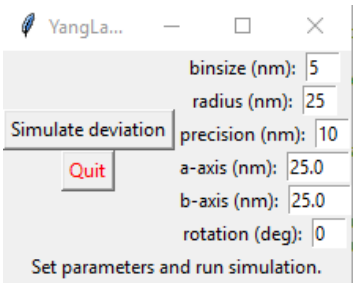


b. Click “Simulate deviation” after entering the proper integer values for the simulation parameters you would like to run. Since the simulation uses 1,000,000 points, the bin size may be set small. The labeling efficiency should be set as a value between 0.0 and 1.0. The rotation should be set as an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

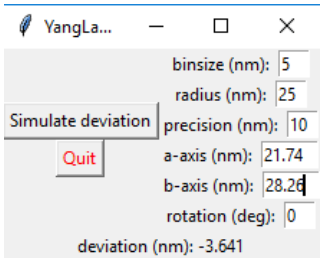


6) Supplemental Figure 11 – symmetry compression

a. Use the same steps as above to open the gui for the “Supplemental Figure 11 - 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.



7) 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.