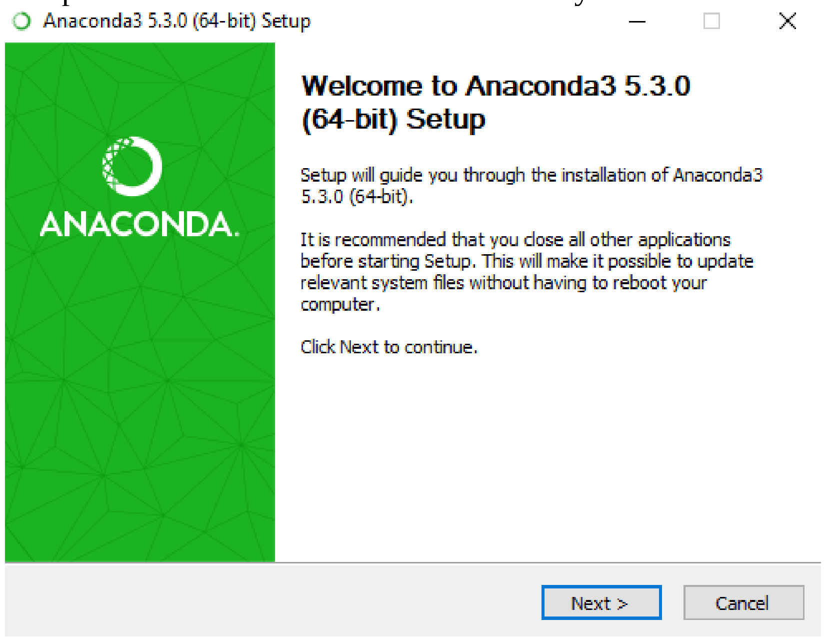
README

Outline:

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.



Welcome to Anaconda3 5.3.0 (64-bit) Setup

Setup will guide you through the installation of Anaconda3 5.3.0 (64-bit).

It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.

Click Next to continue.

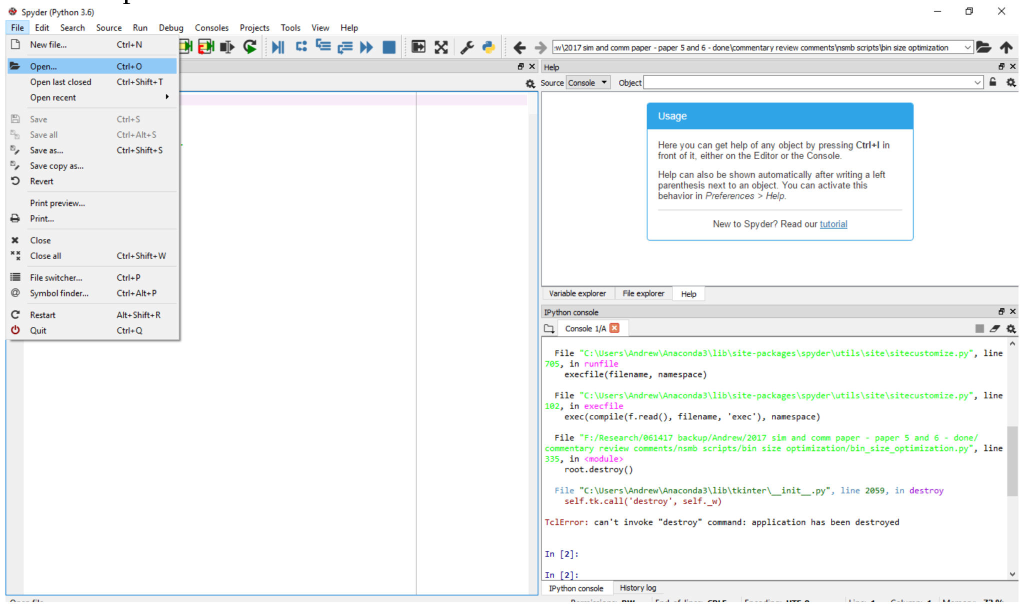
2) Download the script files

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.



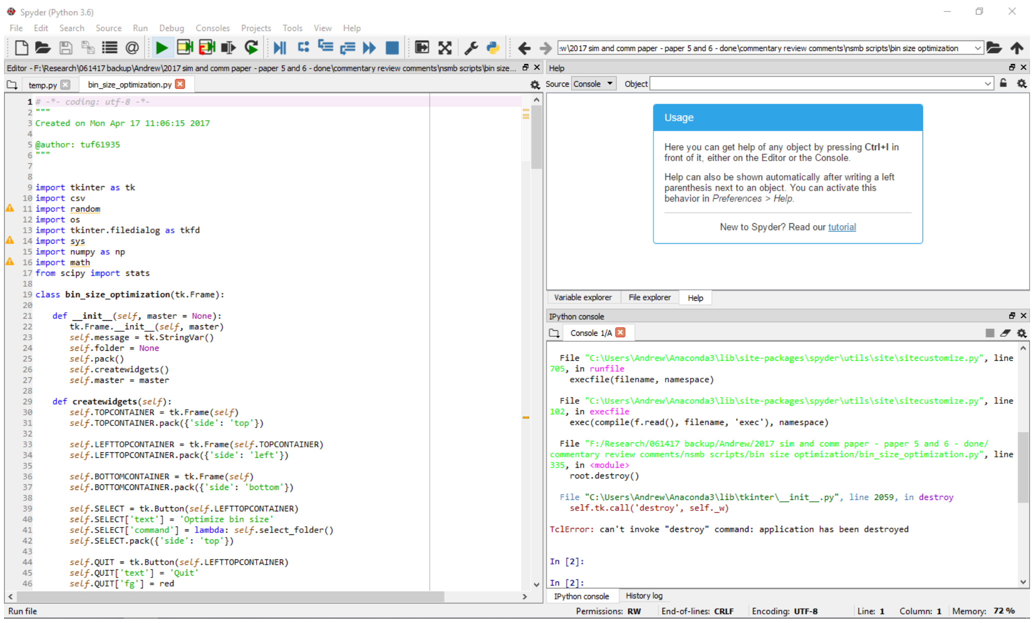
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
  File "C:\Users\Andrew\Anaconda3\lib\site-packages\spyder\utils\site\sitecustomize.py", line
705, in runfile
    execfile(filename, namespace)

  File "C:\Users\Andrew\Anaconda3\lib\site-packages\spyder\utils\site\sitecustomize.py", line
102, in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "F:/Research/061417 backup/Andrew/2017 sim and comm paper - paper 5 and 6 - done/
commentary review comments/nsmb scripts/bin size optimization/bin_size_optimization.py", line
335, in <module>
    root.destroy()

  File "C:\Users\Andrew\Anaconda3\lib\tkinter\__init__.py", line 2059, in destroy
    self.tk.call('destroy', self._w)

TclError: can't invoke "destroy" command: application has been destroyed


In [2]:

In [2]:
```

c. Once "simulation_gui.py" is open, click the green "Run file" arrow in the toolbar.


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - F:\Research\061417 backup\Andrew\2017 sim and comm paper - paper 5 and 6 - done\commentary review comments\nsmb scripts\bin size...
temp.py
bin_size_optimization.py
Help
Source Console
Object
Variable explorer
File explorer
IPython console
Console 1/A
History log
Run file

```python
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 17 11:06:15 2017

@author: tuf61935
"""


import tkinter as tk
import csv
import random
import os
import tkinter.filedialog as tkfd
import sys
import numpy as np
import math
from scipy import stats

class bin_size_optimization(tk.Frame):

    def __init__(self, master = None):
        tk.Frame.__init__(self, master)
        self.message = tk.StringVar()
        self.folder = None
        self.pack()
        self.createwidgets()
        self.master = master

    def createwidgets(self):
        self.TOPCONTAINER = tk.Frame(self)
        self.TOPCONTAINER.pack({'side': 'top'})

        self.LEFTTOPCONTAINER = tk.Frame(self.TOPCONTAINER)
        self.LEFTTOPCONTAINER.pack({'side': 'left'})

        self.BOTTOMCONTAINER = tk.Frame(self)
        self.BOTTOMCONTAINER.pack({'side': 'bottom'})

        self.SELECT = tk.Button(self.LEFTTOPCONTAINER)
        self.SELECT['text'] = 'Optimize bin size'
        self.SELECT['command'] = lambda: self.select_folder()
        self.SELECT.pack({'side': 'top'})

        self.QUIT = tk.Button(self.LEFTTOPCONTAINER)
        self.QUIT['text'] = 'Quit'
        self.QUIT['fg'] = red
```

Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

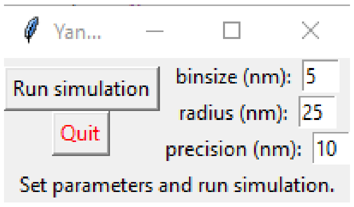
Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
  File "C:\Users\Andrew\Anaconda3\lib\site-packages\spyder\utils\site\sitecustomize.py", line
705, in runfile
    execfile(filename, namespace)

  File "C:\Users\Andrew\Anaconda3\lib\site-packages\spyder\utils\site\sitecustomize.py", line
102, in execfile
    exec(compile(f.read(), filename, 'exec'), namespace)

  File "F:/Research/061417 backup/Andrew/2017 sim and comm paper - paper 5 and 6 - done/
commentary review comments/nsmb scripts/bin size optimization/bin_size_optimization.py", line
335, in <module>
    root.destroy()

  File "C:\Users\Andrew\Anaconda3\lib\tkinter\__init__.py", line 2059, in destroy
    self.tk.call('destroy', self._w)

TclError: can't invoke "destroy" command: application has been destroyed


In [2]:

In [2]:
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 1 Column: 1 Memory: 72 %

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.



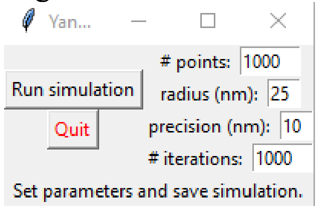
binsize (nm): 5
radius (nm): 25
precision (nm): 10
Set parameters and run simulation.

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.

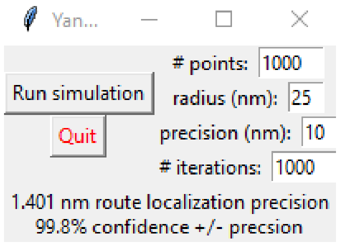
4) Figure 5 – point number simulation

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



points: 1000
radius (nm): 25
precision (nm): 10
# iterations: 1000
Set parameters and save simulation.

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.

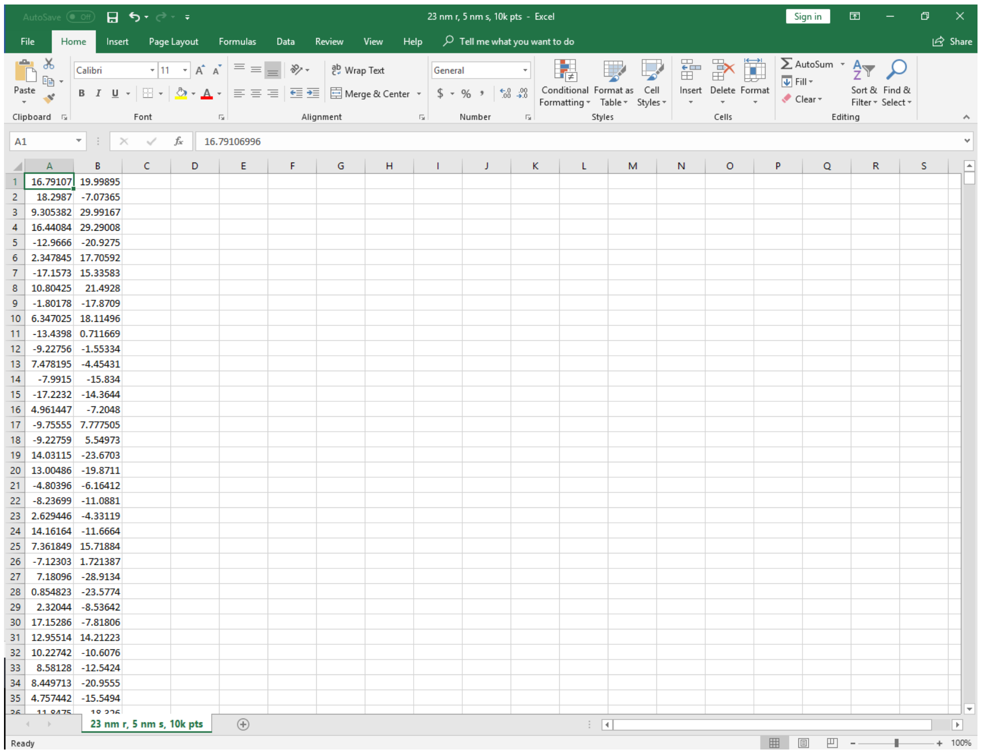


points: 1000
radius (nm): 25
precision (nm): 10
# iterations: 1000

1.401 nm route localization precision
99.8% confidence +/- precsion

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.



A1 — 16.79106996

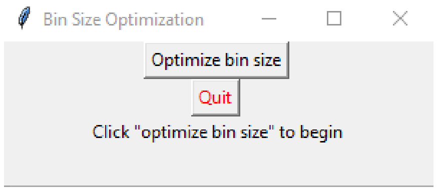
	A	B
1	16.79107	19.99895
2	18.2987	-7.07365
3	9.305382	29.99167
4	16.44084	29.29008
5	-12.9666	-20.9275
6	2.347845	17.70592
7	-17.1573	15.33583
8	10.80425	21.4928
9	-1.80178	-17.8709
10	6.347025	18.11496
11	-13.4398	0.711669
12	-9.22756	-1.55334
13	7.478195	-4.45431
14	-7.9915	-15.834
15	-17.2232	-14.3644
16	4.961447	-7.2048
17	-9.75555	7.777505
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

23 nm r, 5 nm s, 10k pts

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

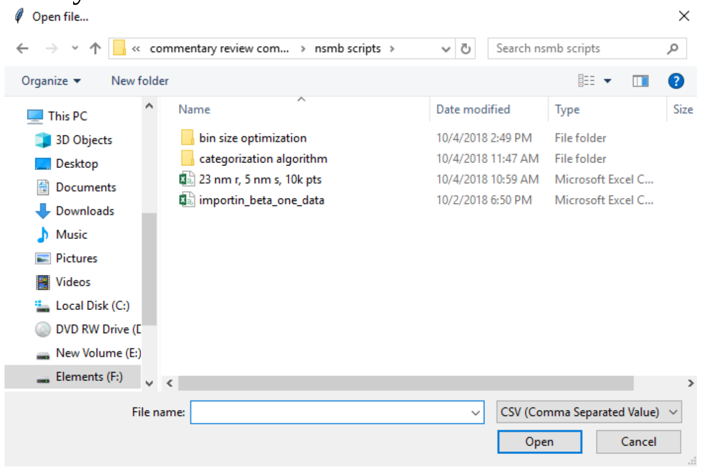
6) Figure 6 – route categorization – bin size optimization simulation

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.



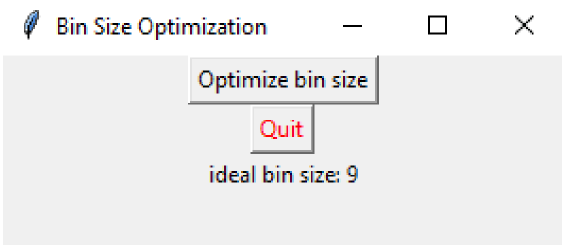
Click "optimize bin size" to begin

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



Name	Date modified	Type	Size
bin size optimization	10/4/2018 2:49 PM	File folder	
categorization algorithm	10/4/2018 11:47 AM	File folder	
23 nm r, 5 nm s, 10k pts	10/4/2018 10:59 AM	Microsoft Excel C...	
importin_beta_one_data	10/2/2018 6:50 PM	Microsoft Excel C...	

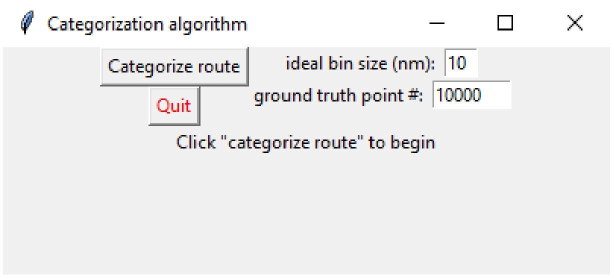
c. After you select the .csv file with your xy data, click “Open”. After a few moments, the optimized bin size will be displayed in the gui window and another .csv file named “binsize_optimization_results.csv” with the raw data for the bin size optimization calculation will be written into the same folder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



ideal bin size: 9

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.



Categorization algorithm

Categorize route

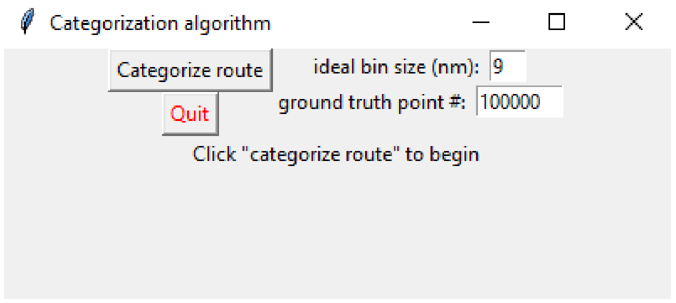
Quit

ideal bin size (nm): 10

ground truth point #: 10000

Click "categorize route" to begin

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.



Categorization algorithm

Categorize route

Quit

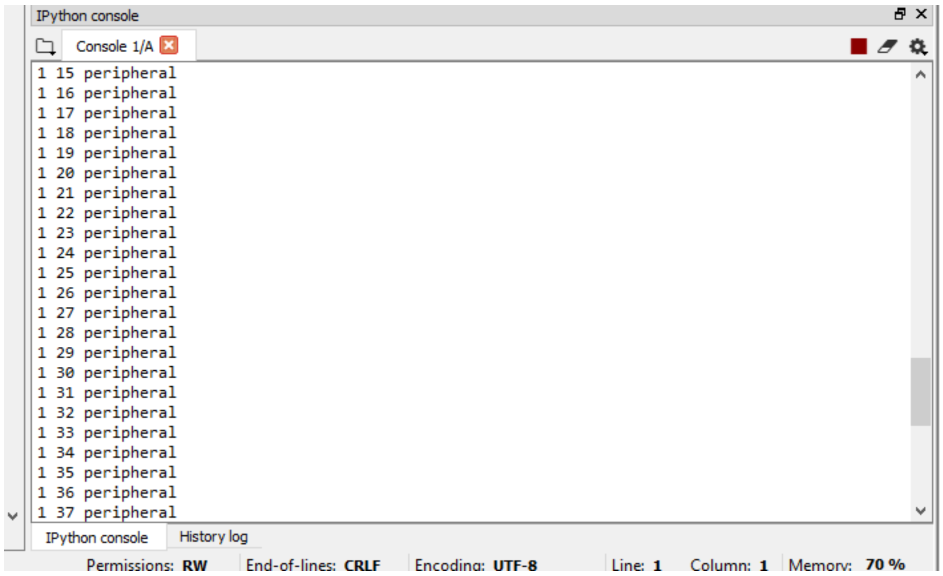
ideal bin size (nm): 9

ground truth point #: 100000

Click "categorize route" to begin

c. Click "Categorize route", navigate to the location of your .csv file with your xy data, and click "Open".

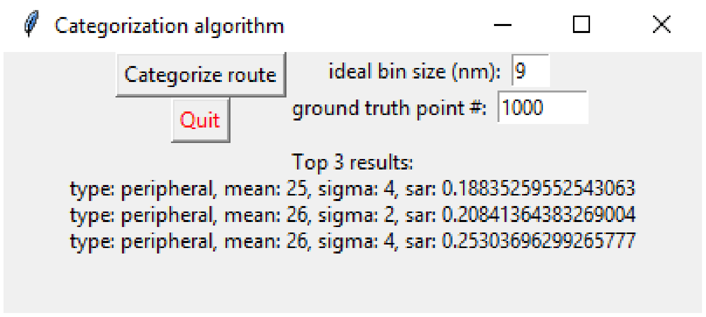
d. At this point the simulation will begin according to the parameters entered in the gui. No other values need to be provided. The progress can be monitored in the IPython console in Spyder. As mentioned above, if ~1,000 points are used to simulate the ground truth distribution, the simulation will take several minutes to complete. On the other hand, if >100,000 points are used, the simulation will need to run for several hours.



```
1 15 peripheral
1 16 peripheral
1 17 peripheral
1 18 peripheral
1 19 peripheral
1 20 peripheral
1 21 peripheral
1 22 peripheral
1 23 peripheral
1 24 peripheral
1 25 peripheral
1 26 peripheral
1 27 peripheral
1 28 peripheral
1 29 peripheral
1 30 peripheral
1 31 peripheral
1 32 peripheral
1 33 peripheral
1 34 peripheral
1 35 peripheral
1 36 peripheral
1 37 peripheral
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 1 Column: 1 Memory: 70 %

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".

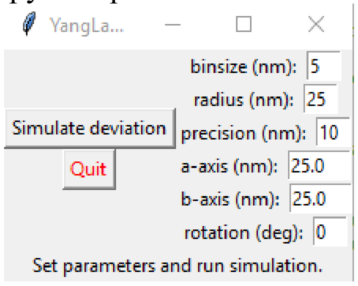


ideal bin size (nm): 9
ground truth point #: 1000

```
Top 3 results:
type: peripheral, mean: 25, sigma: 4, sar: 0.18835259552543063
type: peripheral, mean: 26, sigma: 2, sar: 0.20841364383269004
type: peripheral, mean: 26, sigma: 4, sar: 0.2530369629926577
```

8) Figure 7 – symmetry compression

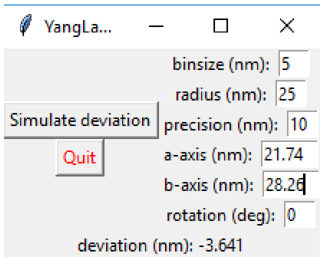
a. Use the same steps as above to open the gui for the "Figure 7 - symmetry compression\simulation_gui.py" script.



binsize (nm): 5
radius (nm): 25
precision (nm): 10
a-axis (nm): 25.0
b-axis (nm): 25.0
rotation (deg): 0

```
Set parameters and run simulation.
```

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.

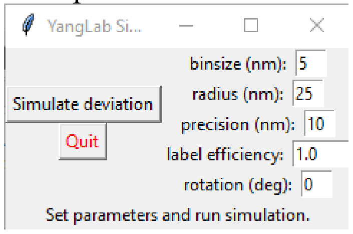


binsize (nm): 5
radius (nm): 25
precision (nm): 10
a-axis (nm): 21.74
b-axis (nm): 28.26
rotation (deg): 0

```
deviation (nm): -3.641
```

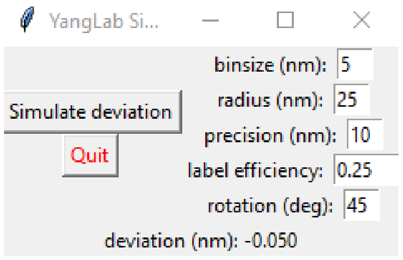
9) Figure 8 – labeling efficiency simulation

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



```
binsize (nm): 5
radius (nm): 25
precision (nm): 10
label efficiency: 1.0
rotation (deg): 0
Set parameters and run simulation.
```

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.

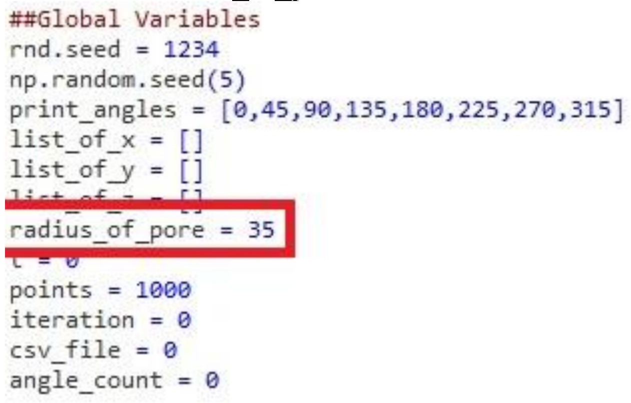


```
binsize (nm): 5
radius (nm): 25
precision (nm): 10
label efficiency: 0.25
rotation (deg): 45
deviation (nm): -0.050
```

10) Figure 9 – fold symmetry and rotation: generate simulated data set

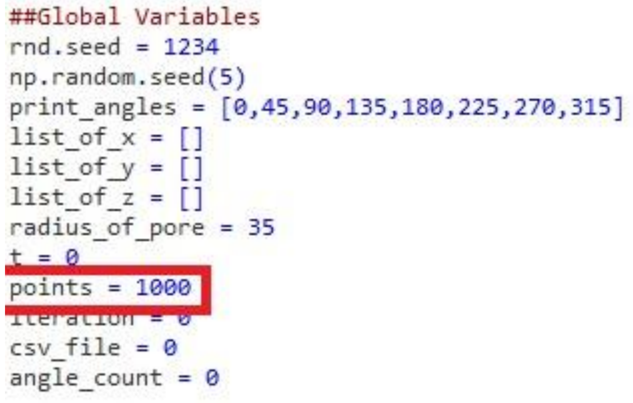
The Point_Circle_Generator program will generate a specified number of points within the bounds of eight circles, called probability lobes. The radius of these circles as well as the degree of localization error is determined by the user. After completing the simulation, the program will move the CenterPoint of the probability lobes by one degree. This will proceed until all possible rotations have been generated. resulting coordinates for each lobe position are saved to a .csv file in a folder titled ‘Simulated_Dataset’.

a. Open Point_Circle_Generator.py in the Anaconda environment.
b. Specify the radius of the circle around which the probability lobes rotate. This variable is called ‘radius_of_pore’.



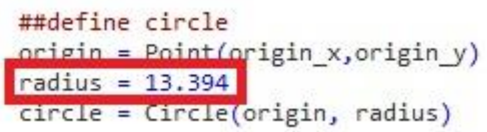
```
##Global Variables
rnd.seed = 1234
np.random.seed(5)
print_angles = [0,45,90,135,180,225,270,315]
list_of_x = []
list_of_y = []
list_of_z = []
radius_of_pore = 35
t = 0
points = 1000
iteration = 0
csv_file = 0
angle_count = 0
```

c. Specify the number of points that each simulated dataset should consist of by changing the variable called ‘points’.



```
##Global Variables
rnd.seed = 1234
np.random.seed(5)
print_angles = [0,45,90,135,180,225,270,315]
list_of_x = []
list_of_y = []
list_of_z = []
radius_of_pore = 35
t = 0
points = 1000
iteration = 0
csv_file = 0
angle_count = 0
```

d. Specify the radius of the probability lobe by changing the variable ‘radius’ found under the define circle section.



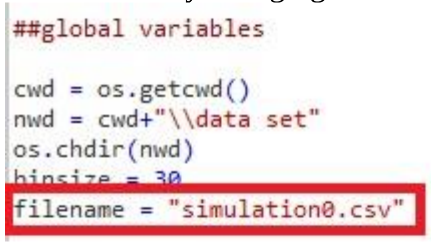
```
##define circle
origin = Point(origin_x,origin_y)
radius = 13.394
circle = Circle(origin, radius)
```

e. Run the program. The simulated datasets will be located within a folder generated by the program called ‘Simulated_Dataset’. Each .csv title consists of two parts. First the word simulation then the degree of rotation applied to the initial degree set. For example, simulation0.csv will be the initial position of the probability lobes. Simulation1.csv will be first rotation, or second generated dataset.

11) Figure 9 – fold symmetry and rotation: 2D to 3D conversion

The 2D-3D converter will convert the 2-dimensional dataset generated by the Point_Circle_Generator program into a 3-dimensional histogram.

a. Copy or move the folder ‘Simulated_Dataset’ generated by the Point_Circle_Generator program into a new folder called ‘Data Set’ in the same folder that the 2D-3D-Converter program file is located.
b. Open 2D-3D-Converter.py in the Anaconda coding environment.
c. Specify which dataset to convert by changing the filename.



```
##global variables

cwd = os.getcwd()
nwd = cwd+"\\data set"
os.chdir(nwd)
binsize = 30
filename = "simulation0.csv"
```

12) 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.