Midterm

PHY411: Do all problems (problem 3 is weighted more kindly in grading).
PHY 506: Do all problems.

Accept the assignment from github classroom: https://classroom.github.com/a/_d2RXL6P. You will then get a link to your own github area.
You should submit your code through github classroom. Submit your writeup, and a link to your github classroom area where your code is, on UBLearns.

Problem 1: Pandemic Zombies

This is intended for educational purposes and is not intended as a realistic simulation of the COVID-19 virus.

Start with python notebook “Midterm/Problem1.ipynb”.

Suppose you have a pandemic of some disease. In this case, we will simulate the zombie apocalypse. As such, we will consider `n_walker` random walkers in 2 dimensions that each take `n_steps` steps. At each step, the walkers can move `dx` units. The walkers are confined to a single 1x1 room with cyclic boundary conditions in x and y (so, the walkers wrap around if they wander off the edge). There is a configurable initial number of infected individuals, `n_infected`.

The code provided assumes that there is zero rate of transmission from the infected individuals (i.e., passive zombies). The animation shows the paths of the walkers. The healthy individuals show up as blue dots. The zombies show up as red dots.

Part a (15 points): Implement an "infection rate" `P_i` and an "infection distance" `d_i` where any walkers that are within the infection distance `d_i` to an infected individual have a probability `P_i` to be infected. They can then infect others in later time steps. The animation should reflect the changes. (Hint: this can’t be vectorized, so you have to use for loops). Assume: `n_walkers=100`, `n_steps=100`, `n_infected=4`, `dx=0.01`, `P_i = 0.1`, and `d_i=0.1`.

Part b (10 points): Plot the number of total infections in the sample as a function of time step for `P_i = 0.1, 0.2, 0.3`, and for `d_i = 0.01, 0.02, 0.03`. Assume: `n_walkers=100`, `n_steps=1000`, `n_infected=4`, `dx=0.01`.

Remember to stay healthy and practice social distancing (keep your distance larger than `d_i` in real life! Remember $d_i = 2$ meters.)

Problem 2: Geospatial location of COVID19 confirmed cases

This is intended for educational purposes and is not intended as a realistic analysis of the COVID-19 virus confirmed cases.

Start with python notebook “Midterm/Problem2.ipynb”.

Part a (10 points): Get the dataset and extract mainland US cases

- Follow directions in the notebook to download the data.
- Extract the data for the mainland US following directions in the notebook. Extract the number of confirmed cases that occurred on 3/18/20 (18-March-2020).

Part b (15 points): Make a Voronoi diagram of the data above.

- Make a Voronoi diagram of the data above.
- Assume there are 10 Voronoi cells. Assign each a different color.
- Initialize them to 10 randomly assigned points in the data sample.
- Compute the centroids that minimize the k-means distance to the data points.
- Plot the centroids of the data in black circles.
- Plot the separate individual points by centroid color.

Problem 3

Start with python notebook “Midterm/Problem3.ipynb”. The “easier” part of the problem is weighted more for undergrads.

Suppose you have a detector response that has a bias $f(x) = \sqrt{0.01^2 + (0.01\exp(-x/4))^2}$. Adjust the "smear" function to be a Gaussian with width 0.03, but with varying bias according to f(x). Given a "true" input Gaussian distribution with mean 0.4 and width 0.1, generate the appropriate measured distribution given the above bias and width. Then unfold that measured distribution and compare to the truth. In effect, you are smearing the true distribution and then unsmearing it to get back where you started ("closure test”).

- **Part a (Undergrads: 15 points. Grads : 10 points):** Plot the “true” distribution and the “measured” distribution after the detector smearing is applied as described above.
- **Part b (Undergrads: 10 points. Grads : 15 points):** Unfold the measured distribution with regularization. Show the regularization optimization, and compare the “best” result to the measured and true values in a plot.