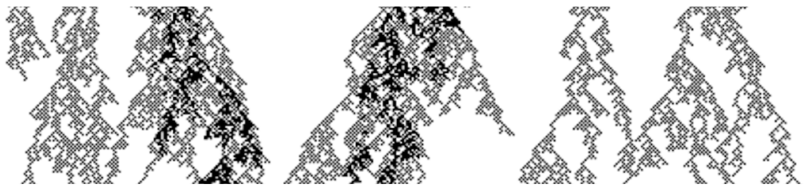
Working with and Visualizing Array-Based Models
DS3000 / DS5110 (Rachlin)



This assignment is designed to help you practice your python programming skills (for those of you who are new to Python. The 2nd and 3rd problems focus on working with numpy arrays which we will cover this week in class.

Problem 1. Further Exploring the Condorcet Model (20 points)

Modify the Condorcet model simulation code given in class (posted to Canvas) so that instead of jurors being assigned a random individual accuracy from 0.50 to some maximum value, they are all assigned the *same* value from 0.00 to 1.00. Plot the curve for the accuracy of the majority based on 10,000 or more random trials as a function of the individual juror accuracy. Overlay three curves for 11, 21, and 31 jurors.

Problem 2. Modeling Infection (30 points)

Imagine N students sitting in a row in a large classroom. Some fraction of the students, P_{sick}, are infected with the 24-hour flu. Model the transmission of the flu from day to day using an MxN array, where N is the number of students, and M is the number of days to be simulated. Who is infected from day to day is determined by the following rules:

a. If a person sick, they are healthy the next day, but may again catch the flu from one of their sick neighbors the day after that. (There is no long-term immunity.)

b. If a person is sick, they transmit the flu to the person on their left or their right each with probability $P_{infect.}$ So, one or both adjacent neighbors may become infected and show up as sick the next day.

Visualize the course of the disease day by day using your MxN array. In the array, 0=healthy and 1=infected. Can you find a value of P_{infect} where the disease persists in the population without simply dying out (everyone becomes healthy or everyone becomes infected?) I recommend you try N=1000, M=1000, P_{sick} = 0.15.

Extra credit (required for DS5110 students): Plot the % of students that are sick as a function of time for three different values of P_{infect}.

Problem 3. The Trees of Boston (50 points)

The file *trees.csv* contains the GPS location of over 200,000 trees in the City of Boston. Create a map of the trees of Boston by mapping the GPS-coordinates of each tree to the (i, j)-coordinates of a 1000 x 1000 array. Each cell in the array should contain an integer representing the total number of trees that map to that specific location. For this exercise, your 1000 x 1000 array should represent the following area encompassing the city of Boston:

Latitude: 42.2 to 42.4 degrees North
Longitude: -71.2 to -70.9 degrees West

Remember that location (0,0) in the array is the North-West corner of the map, and location (999,999) is the South-East corner of the map. (Increasing i-index is *decreasing* latitude while increasing j-index is *increasing* longitude.)

WHAT TO SUBMIT: Your code and all visualizations. (You may embed visualizations in a Jupyter notebook.) DON'T FORGET TO SUBMIT BOTH YOUR CODE AND YOUR VISUALIZATIONS FOR FULL CREDIT. PLEASE VERIFY YOUR GRADESCOPE SUBMISSION TO ENSURE THAT ALL FILES HAVE BEEN POSTED.