M3C 2018 Homework 1

Due date/time: 2/11/2018, 23:59 GMT

This project consists of two parts – in the first, you will develop Python code to simulate a stochastic model of the evolution of two tribes competing for territory. In the second, you will analyze and discuss simulation results.

Getting Started: If needed, update your local copy of the course repo (*sync* and *pull*). The homework/hw1/ directory contains the file, *hw1_template.py*. Make a new directory outside of the course repo, and place a copy of the template file named *hw1_dev.py* in this new directory. Ultimately, the final file that you submit should be titled *hw1.py*. Browse through the *dev* file; there are a number of function headers, and you will have to add code for each of these functions as described below. First, however, you should *add your name and 8-digit college id to the docstring at the top of the file.*

We will consider a model where a *Collaborator* (C) and a *Mercenary* (M) tribe compete against each other for territory. Conveniently, this territory has been divided into a square *N* x *N* grid of villages (where *N* is an odd positive integer greater than *5*)

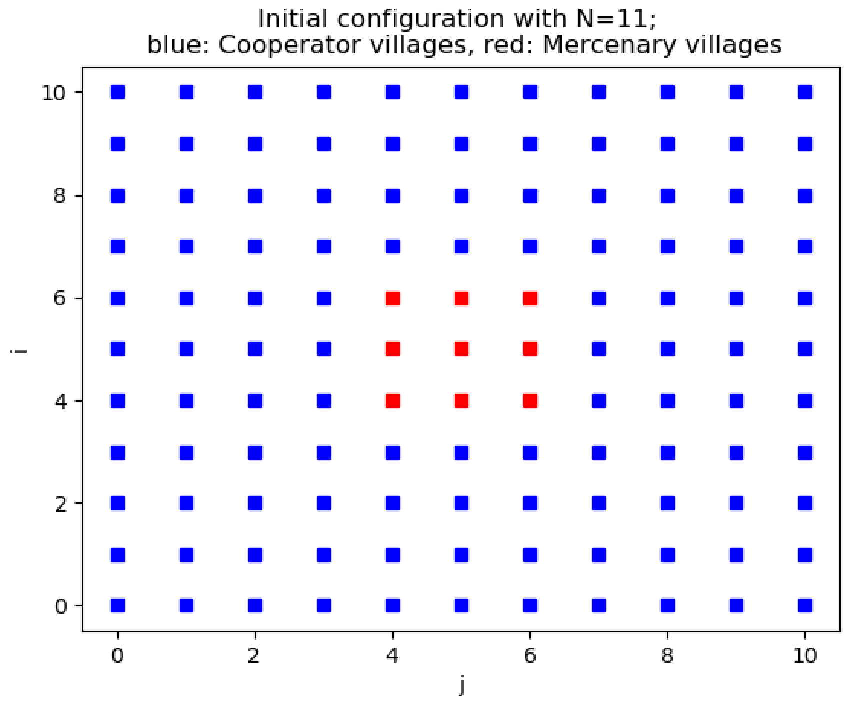
Initially, the central *3* x *3* grid of villages is affiliated with M while all other villages are affiliated with C (see figure below). Each year, a village is given a *fitness* score based on the affiliation of the village and the affiliations of neighboring villages (precise details are provided below). A village changes affiliation after each year with a probability that depends on the fitness values of it and its neighbors.

Each village is assigned a distinct coordinate *(i,j)* with *i* and *j* taking values from $0, 1, \dots, N-1$. The neighbors for village (i,j) consist of the nodes $(i \pm 1, j)$, $(i, j \pm 1)$, $(i+1, j+1)$, $(i-1, j+1)$, $(i+1, j-1)$, $(i-1, j-1)$ and coordinates in this list falling outside of the *N* x *N* grid should be ignored. So, a village may have 8, 5, or 3 neighbors depending on whether or not it is in the interior or on a boundary or corner of the grid.

Each year, a C village receives one point for each neighbor that is also a C (reward for collaboration) and zero points for each M neighbor. An M village receives *b* points for each neighbor that is a C and *e* points for each neighbor that is an M. Here, *b* and *e* are model parameters with $b > 1$ and $0 < e \ll 1$. The fitness score for a village is then: *(total points received)/(number of neighbors)*

The probability of a village being a C the following year is: *(total fitness of C villages in community)/(total fitness of all villages in community)* where the *community* for village *(i,j)* consists of its neighbors and itself. Similarly, the probability of a village being an M the following year is: *(total fitness of M villages in community)/(total fitness of all villages in community)*.

We are interested in simulating and analyzing how the relative proportion of C and M villages evolves over time.



Part 0. (5 pts) You should develop your code in a new git repository. If using Bitbucket, the online repo **must** be private. You should make at least a few commits as you work through the assignment. Once you have completed the assignment, generate a text file, *hw1.txt* at the command line in a Unix terminal which contains the log for your repo. Edit the file so that the first line contains the command used to first generate the file.

Part 1. (50 pts) Complete the function *simulate1* so that it simulates this model with *N*, *b*, *e*, and *Nt* provided as input. Here *Nt* is the number of iterations (years) to run the simulation. The initial village configuration has been provided in the matrix, *S*. The C villages correspond to coordinates with $S_{i,j} = 1$ and M villages correspond to $S_{i,j} = 0$. The function should return the final *S* at the end of your simulation. Additionally, the function should return the array *fc* which contains the fraction of villages which are Cs at each timestep (including the initial time). A simple function for visualizing *S* (*plot_S*) has been provided for you, and you can use (or not use) this as you wish while you develop your code.

Part 2. (45 pts) As *b* is increased (with *e* and *N* held fixed), we expect M to be more and more succesful, and in the second part of the project, you should develop Python code in the function *analyze* which… analyzes if and to what degree this trend is observed. You are encouraged to develop your own approach to this problem, but it may be helpful to consider the evolution of *fc* for different values of *b*. Another point to consider is whether or not *fc* reaches a relatively stable value at long times for some or all values of *b*. Your function should produce 2-4 figures which illustrate the most important qualitative trends which you have identified, and the docstring of *analyze* should contain a clear, concise discussion of these trends and your main conclusions. Save your figures as *.png* files with the names *hw11.png*, *hw12.png*, …, and submit them with your codes. It is sufficient for your final results to focus on *N=21*, *e=0.01*, though you are welcome to present results for other values if you find interesting trends or dynamics. You should start by considering *b* in the range, $1 < b \leq 1.5$ and then modify this range if needed. The code in the *name==main* section of the module should call *analyze* and generate the figures you are submitting.

If you decide to modify your *simulate1* code for part 2, please place the modified code in the *simulate2* function provided and call this function from *analyze* instead of *simulate1*.

Notes:

1. All figures created by your code should be well-made and properly labeled. The title of each figure should include your name and the name of the function which created it.

2. Marking will focus on the functionality of your code (does it run? does it produce the right answer?) and the soundness of your analysis, however points may be deducted for substantial inefficiencies.

3. In order to assign partial credit, markers must be able to understand how your code is organized and what you are trying to do.

Submitting the assignment

To submit the assignment for assessment, go to the course blackboard page, and click on the “Assignments” link on the left-hand side of the page, and then click on Homework 1. Click on “Write Submission” and add the statement: “This is all my own unaided work unless stated otherwise.” If you are a CDT student, add a 2nd line with “CDT Fluids”, or “CDT Neuro” as appropriate.

Click on “Browse My Computer” and upload your final files, *hw1.txt*, *hw1.py*, *hw11.png*, *hw12.png*, etc… . Finally, click “Submit”.