Simulating Human Population Growth and Migration

Karan Shah and Kartikeya Kumar
Georgia Institute of Technology

Introduction

How do human populations behave over time?
Our simulator can be used to simulate population behavior with respect to population growth and migration in a world consisting of different countries.

Conceptual Model

The world is a grid of cells. Each cell has a fixed population carrying capacity and a wealth property. A block of contiguous cells forms a country. Each country has some macroproperties. We generated normal distributions with the value of these macroproperties as means to populate each cell within the country. The macroproperties we used in this simulator were average wealth and initial population distribution. So, cells in a country with higher wealth property will have higher wealth on average.

Population growth: The change in population was modeled by

$$B(N) = b_1 N - b_2 N^2$$
$$D(N) = d_1 N + d_2 N^2$$

where N is the current population. $B(N)$ & $D(N)$ are number of births and deaths respectively. The N^2 terms take care of overpopulation. This can be written as a logistic equation[1]:

$$\frac{dN}{dt} = B(N) - D(N)$$
$$= (b_1 - d_1)N\left(1 - \frac{b_2 + d_2}{b_1 - d_1} N\right)$$

Migration: Let the population of cell i be P_i. We define a function desirability of a cell j to the population of cell i:

$$\delta_{ji} = \frac{1}{d_{ij}} \sum_k w_k F_k(p_{ki}, p_{kj})$$

The function $F_k(p_{ki}, p_{kj})$ gives the difference between the kth property p of cells i and j. At every time step, a small proportion of population in each cell moves to the cell on the grid it finds most desirable.
This is similar to an earlier model[2] using Push Pull factors but it did not take population growth into account.

Implementation

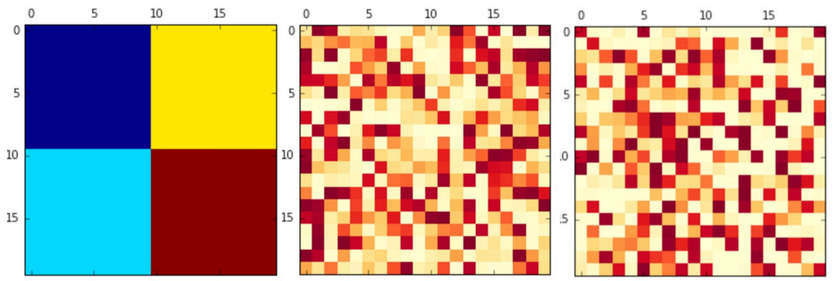
We implemented the simulator in Python in a Jupyter notebook. We used Numpy extensively to manipulate our grid world. All plots were made using Matplotlib.

We wrote our own Gaussian distribution generator to create distributions according to the conceptual model. It was used to populate the grid with people and resources. For population growth, we solved the logistic differential equation numerically.

For migration, we had to come up with an algorithm that checked the desirability of all cells for people in each cell and pick the most desirable destination. A simple brute-force $O(n^2)$ would have been too slow to produce results in a reasonable time frame. So, we implemented our own $O(n \log n)$ recursive algorithm based on Barnes Hut[3] algorithm, which is traditionally used to simulate N-body gravitational problems. This allowed us to run migration simulations for 100s of time steps in a minute.

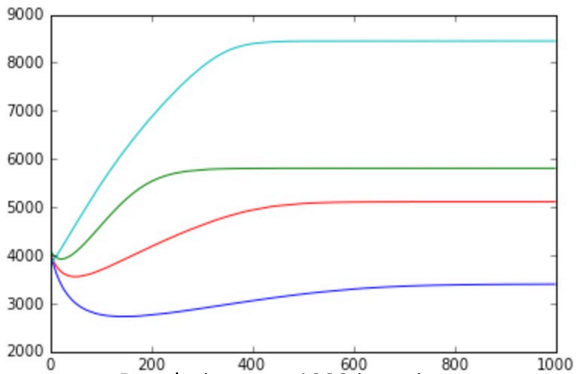
At each time step, our simulator first updates the population and then makes people migrate. Users can control many parameters such as wealth distribution, population density, birth/death rates etc. and see the effects on the population.

Initial Grid

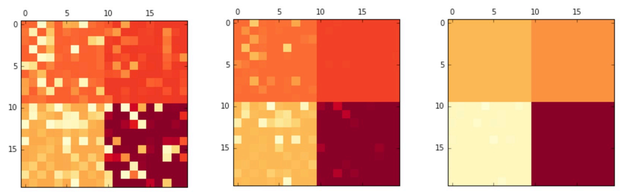


Figures: Left: 20x20 Initial grid divided into 4 countries, Middle: Distribution of population per cell, Right: Wealth distribution

Grid after 1000 iterations



Population over 1000 iterations



Population in cells at t = 200, t = 400 and t = 1000

Limitations

- It can only generate an even number of countries.
- It oversimplifies migration. There can be other factors that affect migration and population growth like elevation, social mobility etc.
- Wealth is a property of each cell and it stays constant. It should change with time and population.
- It can be made slightly more faster by using a full fledged B-H tree.
- Can not be made interactive on account of slow speed.
- Tracking at the level of an individual cell would have been an experiment worth trying.

References

[1]Anon. Logistic population growth. Retrieved April 4, 2016 from http://raven.iab.alaska.edu/~ntakebay/teaching/programming/logistic/node1.html
[2]Guido Dorigo and Waldo Tobler. 1983. Push-Pull Migration Laws. Annals of the Association of American Geographers 73, 1 (1983), 1–17.
[3]Anon. CS267: Lecture 24, Apr 11 1996. Retrieved May 2, 2016 from http://http.cs.berkeley.edu/~demmel/cs267/lecture26/lecture26.html