Laboratory Notebook

Beginning 05-29-2016

JD Russo

Bucknell University

Contents

June 1, 2016

Objective: Refine Gillespie algorithm proof-of-concept, and continue background research.

1 Gillespie algorithm proof-of-concept

After reviewing my first version of my code[1], JJ had feedback on some changes to make, which I'll go through and fix one by one.

$\Rightarrow$ τ should be generated with a different random number than the number used for determining the reaction.

- Changed τ to use a new, freshly generated number.

$\Rightarrow$ An unnecessary division in the code when calculating the propensity array can be removed for efficiency.

- Removed the unnecessary division and timed execution with and without using the `time` command. Too much variation in execution times to determine any significant change.

$\Rightarrow$ How would I modify the array of propensity functions if, for example, reaction 2 were changed from $X \to 2X$ to $2X \to 3X$? Reactions like this can be found in sexual reproduction and chemical reactions.

- I need to look into this more. My first guess would be maybe I need to square the propensity or probability of that reaction, since two reagents are involved.

Before I spend more time trying to answer the last point, I'm going to review some more background materials. I'll start by reading the Wikipedia page on antimicrobial resistance, and go from there.

[1] Git commit `4105445`

June 2, 2016

Objective: Continue background research on antimicrobial resistance and plasmids.

0.1 Background

Antimicrobial Resistance

- Broader topic than just bacteria and antibiotics
- 3 primary ways resistance arises in bacteria
  - Natural resistances
  - Genetic mutation
  - Acquiring resistance from another species **What we're studying with plasmids**
- All classes of microbes develop resistances
- Present in all parts of the world

Mutation

- Low probability of mutating resistances
- Some mutations can produce enzymes that render antibiotics inert
- Some mutations can eliminate the targets of certain antibiotics

Acquired Resistances

- *Conjugation* is passing genetic material from one bacteria to another via **plasmids** and **transposons**. This requires physical contact
- *Transformation* is when a bacteria absorbs genetic material from a free plasmid in its environment
- Viruses can also take genetic material from a bacterium and inject it into other bacteria
- **Horizontal** transfer is sharing genetic material with other bacteria
- **Vertical** transfer is sharing genetic material through reproduction with daughter cells

- **Transposons** are small amounts of DNA that can move between genetic elements like chromosomes and plasmids.

Plasmids

- Cannot reproduce on their own, without a host bacterium
- Widely distributed
- Bacteria can hold many plasmids per cell
- Plasmid genomes have *core* genes for transmission and replication and *accessory* genes that encode traits.

The Plasmid Paradox

"Plasmids impose a fitness cost on their bacterial hosts that generates selection against plasmid carriage under conditions in which plasmid genes do not provide any benefit to the host ... The great irony of the plasmid paradox is that exposure to conditions that select for plasmid-carried genes can also ultimately lead to plasmid loss."

- Recurrent horizontal transfer
- Genes encoded

Skype with Prof. Dong

- For the $2X \rightarrow 3X$ reaction mentioned in yesterday's entry, the propensity array element for the new reaction must be multiplied by two. Since it involves two reagents now, it's twice as probable. *(Why doesn't this make it half as probable?)*
- The theoretical line should follow the solution to the ODE, which in the simplest case is just the exponential.

  This should be plotted on a semi-log plot (meaning logarithmic y-axis) since it's an exponential function. The slope of this line will be the exponent.
- It would be helpful to average the five independent Gillespie algorithm runs that are executed, and compare the average line to the theoretical line.

  However, the Gillespie algorithm has an element of randomness in the time steps it takes. Therefore, between separate runs, times of datapoints will not line up. Without having points with the same time, we can't meaningfully average them. We're not sure what the answer is to this yet, but we're putting it on hold for now.

  We also discussed the central limit theorem, and how if you increase the population size in the simulation, the simulation runs become far closer to the theoretical model. As the population grows, the standard deviation of the mean is reduced, and the Gaussian that describes the distribution of the runs gets narrower.

In biology, the opposite is often the case, where very small populations lead to wide Gaussians, and much noisier data. For example, in our research, if a parent cell has 5 plasmids and produces two daughters, it's likely that the plasmids will be distributed 3 to one, 2 to the other. However, that's just likely, not guaranteed. If this doesn't happen, and if for example one daughter receives all 5 and the other none, this small initial difference can hugely affect the system as it grows.

- We would like to start discussing a journal article every week to become more familiar with the literature surrounding this topic. To that end, I've been provided with a first article to read[1].
- We talked a little about the background research I had done, and a little more in depth about a few of the topics I'd read on. We talked about how cells can lose plasmids without dying, releasing them back into their environment.
- Finally, we discussed next steps for simulation.

Next Steps

Currently[1], the simulation only models the following two reactions

$$X \xrightarrow{\alpha} \emptyset \tag{0.1}$$

$$X \xrightarrow{\mu_2} 2X \tag{0.2}$$

which correspond to death and reproduction respectively. However, in order to move forward, we'd like to expand the reactions considered.

Theory

We can simulate a bacterium transforming by taking in a plasmid from its environment with a new set of reactions

$$X \xrightarrow{\mu_1} 2X \tag{0.3}$$

$$X \xrightarrow{\alpha} Y \tag{0.4}$$

$$Y \xrightarrow{\mu_2} 2Y \tag{0.5}$$

where X is a bacterium that has not absorbed any plasmids and Y is a plasmid carrier. Eqn. 0.4 represents transformation, the process by which a bacterium takes a free plasmid from its environment.

From this, we can quickly see that as per the plasmid paradox mentioned before, $\mu_1 > \mu_2$. In addition, we can determine the cost to fitness of owning a plasmid to be

$$\frac{\mu_1}{\mu_2}. \tag{0.6}$$

[1] Git commit `5932c50`

However, the ratio of X to Y will depend significantly on α, the probability of transformation. We can rewrite the above reactions as differential equations

$$\frac{dX}{dt} = \mu_1 X - \alpha X \tag{0.7}$$

$$\frac{dY}{dt} = \mu_2 Y + \alpha X \tag{0.8}$$

that describe the growth of each population. These have analytical results, but may be manipulated into a form that has an exact solution. I'll need to find this exact solution so I can verify my simulations are producing the proper output.

Once I have the simulation working, I'd like to look at behaviors like how large α can get before Y dominates X.

In addition, α doesn't have to be constant, and in fact should depend on plasmid concentration. I will make it into a function that depends on the plasmid concentration, P.

I'd also like to map the transition of where X and Y dominate. I can do this by plotting $\frac{\mu_1}{\mu_2}$ vs α, and then highlighting regions where each dominate. This will allow us to explore how X and Y change with α.

For now, we will use these 3 reactions and only vary α in order to try and find the $\frac{\mu_1}{\mu_2}$ vs α relation.

In the future, we'd like to perform this simulation on a lattice to take spatial configuration of the bacteria into account, but for now we will assume a well-mixed environment.

June 3, 2016

Objective: Prepare short talk for progress talks today, and begin work on second simulation.

Every Friday we're asked to give a 3-4 minute presentation to the other summer research students. I've spent some time putting together my basic talking points, which can be found in `summer2016/talks/June_3.pdf`.

1 Plasmid conjugation simulation

I'm also starting work on a simulation as mentioned in yesterday's entry, where I replace the initial two (birth and death) reactions with three, bacteria reproduction, transformation, and plasmid carrier reproduction.

I've duplicated the code from the initial simulation to `summer2016/dev/plasmids.py`, where I'll begin modifying it.

I've redefined the reactions, and made the program keep track of X and Y (bacteria and carrier) populations separately.

I can plot the results of this simulation, and see that if I increase alpha, the point where the X and Y populations cross shifts to the left.

June 6, 2016

Objective: Expand plasmid carrier population simulation to run through our parameter space and meaningfully plot results.

1 Plasmid conjugation simulation

Our goal with the plasmid carrier population simulation is to be able to make a plot of $\frac{\mu_1}{\mu_2}$ vs α. I think it would make sense to show this as a contour plot. I can generate a large number of datapoints throughout the plot, and then plot them indicating whether the X or Y population dominated (maybe blue-red spectrum?).

Alternatively, I can save the results of each run (i.e. $\frac{\mu_1}{\mu_2}$, α, and X and Y populations) and then do some interpolation to get a smoother plot, but that may be more effort than it's worth and I'm not sure that's the results we'd even want.

I've modified the code to generate either a contour plot, or an animation of the contour plot at different timesteps. These timesteps are currently independent, i.e. if you run for T = 1, 2, 3 it will not be snapshots of the same simulation at T=1, 2, and 3. It will be 3 separate, independent simulation runs.

Now that I have my simulation results, I will model what plot I'd expect to see from the differential equations describing the reactions.

I've modelled $\frac{\mu_1}{\mu_2}$ vs α in Mathematica, using the same value of $\mu_1 = .8$ and ranges of $\alpha = [.2.., 7]$ and $\mu_2 = [.29....8]$. However, the contour animation in Mathematica does not seem to show the Y population dominating the X as quickly as the simulation does. Examples at $T = 0.9$ and $T = 1.5$ are provided below.

These figures are a little hard to compare though, because the color schemes aren't quite the same. I'll work on finding a way to match those up, or perhaps revisit the interpolated contour plot I have in `plasmids.py`. Using that, I could label the contours on both, and compare values that way.

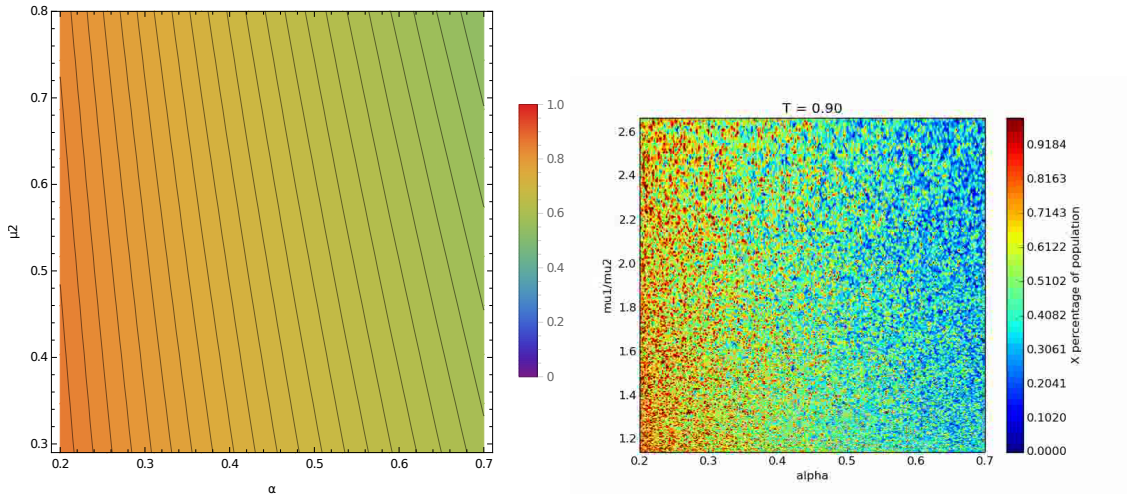

Figure 1: Plots of $\mu 2$ vs α at $T = 0.9$ in Mathematica (left) and Python (right)

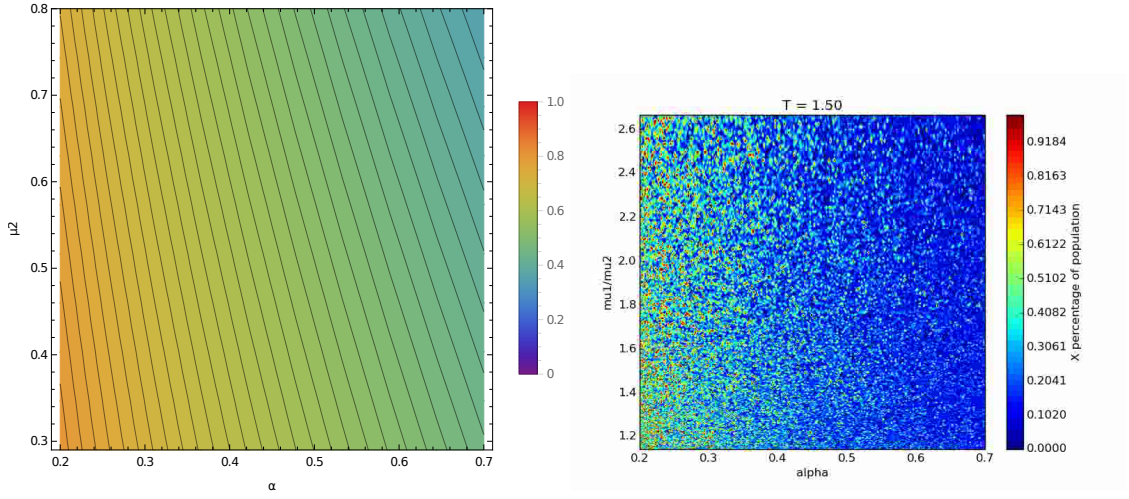

Figure 2: Plots of $\mu 2$ vs α at $T = 1.5$ in Mathematica (left) and Python (right)

June 7, 2016

Objective: Present simulation results to JJ, and consider next steps.

1 Plasmid conjugation simulation

I've added `animate2` and `animate3` function to animate an interpolated contour plot and a 3D surface plot, respectively.

The code will occasionally error if one population goes to zero - handle that more gracefully!

June 8, 2016

Objective: Derive equations for susceptible and resistant population growths, and compare to my results from Mathematica.

We start with the equations

$$\frac{dS}{dt} = b_S S - \alpha S \tag{0.9}$$

$$\frac{dR}{dt} = b_R R + \alpha S \tag{0.10}$$

describing two populations of particles, S and R, where b_S is the birth rate of the S (susceptible) population, b_R is the birth rate of the R (resistant) population, and α is the rate at which an S particle can acquire a plasmid and become an R. These are similar to equations (0.7) and (0.8) mentioned earlier. My goal is to obtain an exact solution for $S(t)$ and $R(t)$.

By rewriting (0.9) as

$$\frac{dS}{dt} = (b_S - \alpha)\, S$$

one can see that there is a solution of the form

$$S(t) = S_0\, e^{(b_S - \alpha)t}. \tag{0.11}$$

$$\frac{dR}{dt} = b_R R + \alpha S \qquad \text{Substitute in (0.11) for S.}$$

$$\frac{dS}{dt} = (b_S - \alpha)\, S \qquad \text{Set } R(t) = y(t)e^{\mu t} \text{ and } \mu = b_S - \alpha.$$

$$\mu y(t)e^{\mu t} + \dot{y}(t)e^{\mu t} = b_R y(t)e^{\mu t} + \alpha S_0 e^{\mu t} \qquad \text{Simplify.}$$

$$\mu y(t) + \dot{y}(t) = b_R y(t) + \alpha S_0 \qquad \text{Solve for } \dot{y}.$$

$$\dot{y}(t) = (b_R - \mu) y(t) + \alpha S_0 \qquad \text{Set } a = (b_R - \mu) \text{ and } b = \alpha S_0.$$

$$\dot{y}(t) = a y(t) + b \qquad \text{Let } x = \dot{y}.$$

$$x = a y + b \qquad \text{Take the derivative } \dot{x}.$$

$$\dot{x} = a \dot{y} \qquad \text{Substitute in } \dot{y} = x.$$

$$\dot{x} = a x \qquad \text{Write the general solution to } x.$$

$$x(t) = x_0 e^{at} \qquad \text{Expand.}$$

$x_0 e^{(b_R-\mu)t} = ay + b$	Solve for y
$y = \frac{1}{b_R - \mu}\left(x_0 e^{(b_R-\mu)t} - \alpha S_0\right)$	Substitute back to get an expression for $R(t)$.
$R(t) = \frac{e^{\mu t}}{b_R - \mu}\left(x_0 e^{(b_R-\mu)t} - \alpha S_0\right)$	Simplify.
$R(t) = \frac{1}{b_R + b_S + \alpha}\left[x_0 e^{b_R t} - \alpha S_0 e^{(b_S-\alpha)t}\right]$	To determine x_0, first solve for $R(0)$,
$R(0) = \frac{1}{b_R + b_S + \alpha}\left[x_0 - \alpha S_0\right]$	then solve for x_0.
$x_0 = (b_R - b_S + \alpha)R(0) + \alpha S_0$	Plug back in to R(t) and simplify.
$R(t) = R_0 e^{b_R t} + \frac{\alpha S_0}{b_R - b_S + \alpha}\left(e^{b_R t} - e^{(b_S-\alpha)t}\right)$	**Exact equation for $R(t)$.**

This is consistent with my results from Mathematica.

June 9, 2016

Skype with Prof. Dong

- We discussed that it would be more useful to view absolute S population, rather than plotting a relative percentage.
- It'd be useful to characterize fluctuations by plotting trajectories for $X_0 = 1000$, $X_0 = 1000000$, and the model.
- We'd like to look at longer timescales, to see how long until X dies out or until Y dominates. The issue with this is the time it takes to run longer simulation.
- The X death timescale should be $\mu_1 - \alpha$.
- We'd like to use a Hill function for α to add some dependence on X_0, since realistically the number of available plasmids would decrease as the population increases and uses them up.
- Michaelis-Menten kinetics are used to relate reaction rates to substrate concentration. This is essentially what we'd be doing with Hill functions and alpha. (Are we doing the reverse, relating reaction rate to reactant concentration?)
- We'd also like to calculate equilibrium conditions for S and R populations (see Michaelis-Menten equilibrium approximation)

- In bacteria populations, the **carrying capacity** of the environment determines the upper limit on the population – we'd like to include this.
- This would change the DE for population growth to

$$\frac{dS}{dt} = b_s \left(1 - \frac{(S+R)}{k}\right) - \alpha S \tag{0.12}$$

  where k is the environment's carrying capacity, the maximum number of individuals it can sustain. This is logistic growth.
- $\frac{dS}{dt} = b_s \left(1 - \frac{(S+R)}{k}\right) - \alpha S$ has solutions at $S = 0, k$. If we start somewhere in between, what determines which equilibrium point we will go to?
- We can see this with Kinetic Monte Carlo by doing time traces at different X_0s and seeing which solution it goes to.

- We'd also like to incorporate a death rate, and calculate the equilibrium point with that.

June 13, 2016

Objective: Refactor code to work more reliably, intuitively, and efficiently.

The first version of `plasmids.py`[1] did not accurately track one population of bacteria to explore time evolution of a system. Instead, it would do an independent simulation to each time step, and plot the results of that. We don't want that, we want to track one simulation to examine the dynamics in it, so I factored that in when rewriting.

I've committed my rewrite[2], which addresses this issue in addition to making a slightly cleaner method for plotting data.

[1] Git commit `bd8b069`
[2] Git commit `3034cd6`

June 13, 2016

Skype with Prof. Dong

We began by reviewing some of my generated plots, and detailing how we could improve them.

Objective: Use equations for $S(t)$ and $R(t)$ to find when $S = R$. This will help us find an interesting parameter range to use.

By setting $\frac{S(t)}{R(t)} = 1$ and solving for t, we can find that the crossover time where $R = S$ is given by

$$t = \frac{1}{b_R - b_S + \alpha} \ln\left(\frac{b_R - b_S + 2\alpha}{\alpha}\right). \tag{0.13}$$

This can be analyzed to find interesting parameter ranges, where the crossover time is within the time spans we're looking at. Until now, I've typically been using values of $\mu_1 = 0.8$, $\mu_2 = .7 to .78$ and $\alpha = 0.1 to 0.1$. However, these yield a crossover time of $32.4 - 5$. This is much longer than I've let my simulations run (because of time constraints), and so I would've never seen this behavior.

By changing α to .1 to .3 this time scale changes to $t = 9.33$ to 2.55, a more reasonable value that is within the timescales we're looking at.

1 Plasmid conjugation simulation

Multiprocessing

As we start to move up to more combinations of parameters, longer timesteps, and more simulation runs, computation time becomes much more important.

In order to make the simulation a little faster, I've implemented multiprocessing[1]. So, when combinations of parameters are run, the program will break off each Gillespie algorithm run to its own process.

System	Parallelized	S_0	T_max	Runtime(s)	Gillespie Runs
Local	Y	100	10	00.64, 00.59, 00.52, 00.57	1
Local	Y	100	15	09.17, 12.55, 09.40, 08.94	1
Local	Y	100	16	16.82, 14.85, 18.12, 18.85	1
Local	Y	100	17	45.51, 31.73, 37.71, 38.26	1
Local	Y	100	18	71.21, 79.51	1
Remote	Y	100	16	64.392	9
Local	Y	100	16	240	9
Local	N	100	16	757	9

[1] Git commit `2380518f`

Not shown, the linux remote is roughly 1/2 as fast for a single run as my local system.

June 20, 2016

- Added plots of R population * Now focusing mostly on X vs T, have found interesting param range
- Added some sanity checks in code for making sure the populations are positive before removing members from them.
- Changed from log base e to log10

June 28, 2016

Objective: Clean up results and prepare presentation quality visualizations of data from simulation.

I'm essentially done with the code for simulating the well-mixed case, so I'm going to take some time to collect my results from the simulation. I will be collecting data and plots of population vs time of both asymmetric and symmetric cases for the following:

- Constant α
- Linear α
- Recycling α

I'd also like to add code to plot the total number of plasmids.

The plots for recycled and constant alpha seem the same.

June 29, 2016

Objective: Separate plotting code from simulation code

June 30, 2016

Objective: Begin planning for simulation on a lattice

July 8, 2016

Objective: Write differential equations for the linear α and recycled α cases.

We can write the differential equations governing the S and R populations with a constant transformation rate α, S and R birthrates μ_1 and μ_2, R death rate δ and a carrying capacity K using the following equations:

$$\frac{dS}{dt} = \mu_1 \left(1 - \frac{S+R}{K}\right) S - \alpha S \tag{0.14}$$

$$\frac{dR}{dt} = \mu_2 \left(1 - \frac{S+R}{K}\right) R + \alpha S - \delta R \tag{0.15}$$

In order to model a population with an initial fixed number P_0 of available plasmids and a linear dependence of α on number of available free plasmids, a third equation must be added to model the number of plasmids. The first two equations must also be revised.

$$\frac{dS}{dt} = \mu_1 \left(1 - \frac{S+R}{K}\right) S - \alpha \left(\frac{P}{P_0}\right) S \tag{0.16}$$

$$\frac{dR}{dt} = \mu_2 \left(1 - \frac{S+R}{K}\right) R + \alpha S \left(\frac{P}{P_0}\right) - \delta R \tag{0.17}$$

$$\frac{dP}{dt} = -\alpha P \tag{0.18}$$

These all assume symmetric divison for R cells, in other words that $R \to 2R$. However, to conserve number of plasmids, this reaction is actually the asymmetric: $R \to S + R$. To include this in the equations for the S and R populations, a term must be moved from the R to S equation.

$$\frac{dS}{dt} = \mu_2 \left(1 - \frac{S+R}{K}\right) R + \mu_1 \left(1 - \frac{S+R}{K}\right) S - \alpha \left(\frac{P}{P_0}\right) S \tag{0.19}$$

$$\frac{dR}{dt} = \alpha S \left(\frac{P}{P_0}\right) - \delta R \tag{0.20}$$

$$\frac{dP}{dt} = -\alpha P \tag{0.21}$$

July 25, 2016

Objective: Revisit lattice simulation.

Now that I've gotten good results for the well-mixed case, it's time to revisit the simulation on a lattice. First, I want to characterize how long my code takes to run as-is, since that could be a large issue. If there's some huge inefficiency, I'd like to be able to start thinking about that early on as I design the simulation, instead of needing to go back and fix it.

A simulation on a 20x20 to t=3.0 took a little over two minutes, or about 19 seconds without live plotting. The same simulation with a 50x50 lattice without live plotting had a runtime of 36 seconds, 87s for a 100x100, and for a 200x200.

My most immediate goals for the lattice simulation are to

- Add arrays to track S, R, and P occupied lattice sites
- Have animation actually update title with current time
- Characterize runtimes

I made a fit to the runtimes which came out to $12.4748 + 0.00824437x$ where x is the total number of lattice sites.

By making a few changes to how an occupied lattice site is selected, I was able to reduce the ticks per call of get_occupied by about 100x.

July 26, 2016

Objective: Continue work on lattice simulation.

Goals for today:

- Keep track of P population
- Add integer counters for N_s and N_r for efficiency (avoid having to recalculate every loop). This may not be necessary if this isn't causing a significant performance hit.

JJ has also brought me N̈onlinear Dynamics and Chaosb̈y Steven Strogatz.

July 27, 2016

Objective: Continue work on lattice simulation, and update population equations to increment total number of available plasmids whenever a symmetric R division occurs.

I've added a few functions to clean up the simulation, and handle adding/removing members from the lattice better. I ran a simulation to $T_max = 10$ using a 50x50 lattice, $\delta = 0.3$, $\alpha = 60, \mu 2 = 0.75$, $S_0 = 500, R_0 = 5, P0 = 2e4, K_site = 5$, under the linear α case. This produced an interesting transition, where you can watch the R boom until the plasmids are depleted, then die out as S takes over. This took about 3 minutes to run.

I'm trying another run with the same parameters, but with a 100x100 lattice and $P0 = 5000$. This took significantly longer, about 17 minutes

I'm storing interesting output animations in the 'dev/videos' directory. I'm also doing some basic profiling of my code, to see if I can help optimize it a little. Since the lattice is so much more complex than the well-mixed case, it naturally carries a fairly significant performance hit with it. Mitigating this has already given me speedups of a few hundred times, but I'm still doing a lot of array searches all over. On this topic, switching from plcolor to imshow for the heatmap was a huge improvement in speed.

Bibliography

[1] Mukund Thattai, Alexander van Oudenaarden *Intrinsic noise in gene regulatory networks* PNAS vol. 98 no. 15 2001. http://www.pnas.orgycgiydoiy10.1073ypnas.151588598/