

CS341 SYSTEMS BIOLOGY I

LECTURE 2

1. ADMINISTRATIVE

- HW #2 is posted. I have a Jupyter Notebook for the coding and a Google Form for the reading. Submit the Google form with the submit button. Email your Jupyter Notebook to me.
- Be sure to fill out the form regarding choice of partners.

2. TALK ABOUT PAPERS READ FOR HW1

- Reading : Biology Textbook
 - DNA (all the genes), gene (part of DNA that encodes a single protein), protein (folded chain of amino acids that ultimately has a job in the cell)
 - central dogma: Gene $\rightarrow$ mRNA via transcription $\rightarrow$ protein via translation
- Reading 2: Computational Systems Biology by H Kitano, Nature, 2002
 - We need to understand how the components interact. Example p53 does not work alone to be a tumor suppressor
 - Robustness is the ability of a system to maintain a specific behavior in the presence of perturbations (and usually, we talk about typical perturbations, like wind, rather than extreme perturbations, like tornados). I might talk about how the circadian is a robust time-keeper in the presence of mild changes in lighting conditions (e.g. clouds don't mess up our ability to keep time during the day).
- Reading 3: Systems Biology: A Brief Overview by H Kitano, Science, 2002
 - The ideal cycle of systems biology involves computation (make a model), analysis (analyze the model), technology (the model should drive questions that require further experimentation), and genomics (analyze the experiments, to learn more, which allows us to change our models to ask and answer new questions).
 - How can modeling improve efficiency? Here is the student answer that Stephanie resonated with the most.

“Modeling is important in identifying plausible, and therefore potentially fruitful, explanations for biological systems. While the models themselves do not prove these explanations to be true, experimental evidence which supports the model can”

3. PREP FOR HW2 PAPER-READING AND HW

- Revisit previous readings. These are very boring videos by Stephanie. She recommends watching them at double-speed.
- Reading 1: Can a biologist fix a radio: or what I learned while studying apoptosis by Lazebnik, 2002
 - This is a fun paper relating how biologists had to study biological systems in the past, but with the vision that they were using those methods on a human-made item. It illustrates how hard it is to reverse-engineer.
- Reading 2: The tale of neuroscientists and the computer by Brown et al., 2014
 - This updates Lazebnik's paper to apply to neuroscience.
- This HW also has a coding component. That will be a jupyter notebook that I would like you to email to me.

4. LEARN HOW TO USE JUPYTER NOTEBOOKS

A Jupyter notebook is a browser-based application that allows you to run Python code and have its output be in the same notebook. The document is broken up in to cells - each cell can have either code or text. If it has code, then once you run the cell, the output becomes part of the document.

There are two options for running the notebooks:

- (1) Install it on your computer Once you have jupyter installed on your computer, you need to start up the server. from a directory that contains (or will contain) your notebook files (.ipynb files). From Terminal, I navigate to my CS341 directory, and launch the server:

```
jupyter notebook
```

That will open a browser. From that browser, I navigate to my homework sub-directory for notebooks associated with homework or my projects sub-directory for notebooks associated with projects. Treat the browser like an IDE (save your file frequently!) and run it in the browser.

- (2) You can use the site Colby has set up for us:
 - <https://cs251.jupyter.colby.edu>
 - You will need to download .ipynb files from the course website, then upload them to this server.

3.78

I write small chunks of code in each cell, then move on to the next cell. That way, I test as I go. I also separate code that computes things (e.g. runs a simulation) from code that plots. I do this because the computation typically takes awhile, and I usually want to tweak my plotting code, so I run the computation once, then plot, readjust the plot code, re-plot, readjust the plot code, etc.

You can import libraries (e.g. `numpy`) and other files (e.g. `.py` files in the same directory). This makes it possible to use the notebook as the top-level code. It can rely on a pretty sophisticated set of libraries. Cool, eh?

5. LEARN KEY FEATURES OF NUMPY, SCIPY, AND MATPLOTLIB

5.1. Numpy. Numpy provides most of the functions we need for storing multi-dimensional numeric arrays and doing computations with them (e.g. dot products, subtracting one vector from another, finding the mean).

Check out the Jupyter Notebook for key links and examples.

5.2. Scipy. Scipy provides more sophisticated numerical functions. We use Scipy to simulate our models (solve ODE's).

We are going to begin with the simplest model typically used. In this model, we have one protein P , whose concentration we will model over time. The process we will model is called degradation - the process by which the protein is destroyed. At any given moment in time, a given proportion of that protein will be destroyed. If we start with a lot of protein, then early in the simulation we will be losing lots of it (it will be degrading quickly). Once we have just a little protein, we lose less at any given moment, so the process is slower. We capture that with the ordinary differential equation

$$\frac{dP(t)}{dt} = -kP(t)$$

where k is the constant that determines the proportion of protein P that is degraded. I read this as "the rate of change of P depends is proportional to the amount of P " and, if we assume that $k \geq 0$ (which is usually the case in these models), then we see that the rate of change of P is negative, and will be more negative (bigger) when P is bigger".

In the Jupyter Notebook, we use `scipy.integrate.solve_ivp` to simulate the degradation model when $k = 2$, $P(0) = 3$, and time is an ndarray from 0 to 10 with a step of 0.1.

5.3. Matplotlib. Matplotlib provides the plotting functions we will use. In the Jupyter Notebook, we demo how to plot a time series (that is a series of values over time).

6. WORKING WITH PARTNERS IN PROJECTS

After each project, you will (individually) fill out a Google Form assessing how well your partnership functioned. Here are the guidelines for working with a partner. We will discuss/edit/expand them in class.

- Neither of us will work on the project without the other's permission.

- The documents with which we are working are equally accessible to both partners. We will share the documents by (Note from SRT: You may want to use Drop-Box, since GoogleDocs isn't going to work well for Jupyter Notebooks unless you have Colaboratory installed, but when Stephanie tried, she found she didn't have permission. Another alternative is using Github to store the notebook and Jupyter Notebook viewer to view it.)
- For each project, we will specify how we plan to split up the work.
- We will take turns taking the lead on every other project. By "taking the lead", we mean...
- We will make sure we listen to each other's ideas.
- We will fulfill our tasks, but if something goes wrong (e.g. an injury), we will tell our partner, so that alternate arrangements can be made.