

Dismiss

Join GitHub today

GitHub is home to over 28 million developers working together to host and review code, manage projects, and build software together.


Sign up

Branch: master ▾

[INF674](#) / 03-Competitive-Epidemics-TP.ipynb

Find file

Copy path

 [balouf](#) Typo

a045bb7 on 2 Oct 2017

[1 contributor](#)

531 lines (530 sloc) 17.6 KB

INF 674 S3: Epidemic Competition

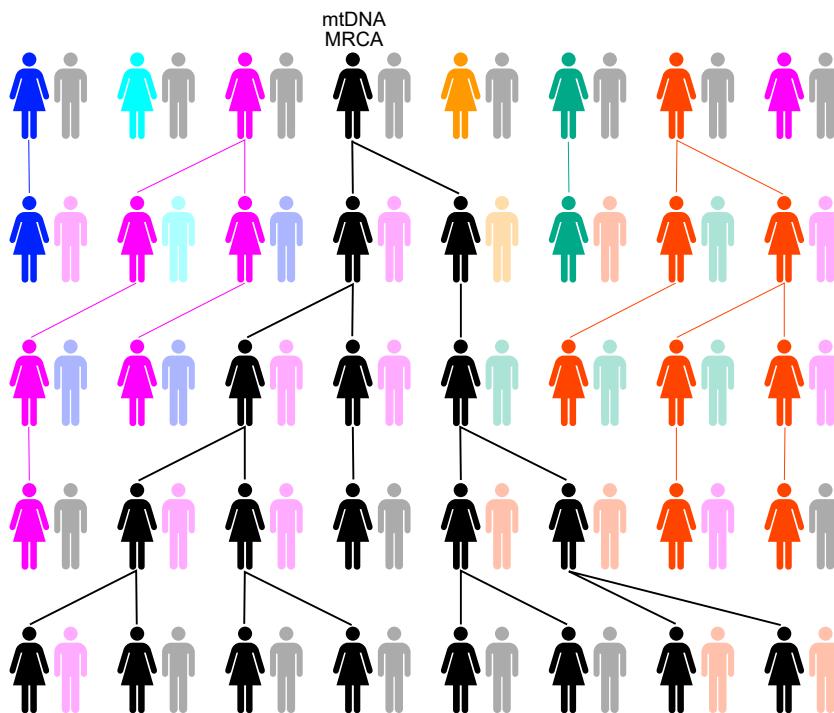
Céline Comte & Fabien Mathieu

2017-2018

Until now, we have dealt with contamination with only one type of *virus*. The only bottleneck was the ability of a node to contaminate another one. For this session, we propose a selection of competition models, where multiple *viruses* compete for a same resource. The goal is not to be exhaustive (there are basically as many competition models than there are researchers working on that field), but to let you see that a same idea can lead to multiple approaches and applications.

1. The Mitochondrial Eve

The context, origin and details of the model will be explained during the course. To be self-contained, we remind the principle: there is a first generation of n individuals (the potential eves). At each next generation, there are n individuals as well. The parent of each individual is chosen uniformly i.i.d. among the individuals of the previous generation. The game stops when one individual of the first generation is the unique ancestor of the current generation: Eve has been found. We propose to study the age of Eve, i.e. the number of generations required for all individuals to share the same Eve. For example, in the picture below, the age of Eve is four.



Question 1

Propose a function `age_of_eve` that returns for a given n the age of Eve computed on one instance.

Hints:

- You're not asked to build the whole family tree, just to compute the age of Eve. Do not compute what you do not need.
- Going forward or backward in time?

```
In [1]: %pylab inline
```

Populating the interactive namespace from numpy and matplotlib

Answer:

Question 2

Compute and display the age of Eve as a function of n averaged over a few trials. Recommended values for a good display: 100 100 2000 100

Compute and display the age of Eve as a function of n , averaged over a few trials. Recommended values for a good display: 100,000,000, 100 trials. When testing your code, you may choose lower values.

Answer:

Question 3

Try to propose some guesses for the expected age of Eve. You decide your methodology. You can try to propose a value according to your experimental results, or try to infer some bounds. When the number of distinct ancestors is large enough, you can estimate the proportion that disappears going up one generation. When only few distinct ancestors remain, you can estimate the number of generations required in average for having one distinct ancestor less. You can Google some bibliography and explain what you found...

Answer:

Question 4

Mitochondrial Eve is estimated to have lived between 99,000 and 300,000 years ago. Population size at that time was about 500,000. Comment the Mitochondrial Eve model we exposed in this session.

Answer:

Question 5 (bonus)

Try to optimize your code.

Python is not designed to be the fastest language of the world, but that does not mean you shouldn't try to write fast functions. Ideally, your function is fast by design, but sometimes you overlooked a possible computational bottleneck and your execution time goes sky high.

The **cProfile** module is a good way to investigate the bottlenecks of your code.

For example, you may investigate your solution for Question 2 by doing something like that:

```
In [10]: import cProfile
cProfile.run('age_of_eve_trials()', sort = 'tottime')

33857749 function calls in 202.858 seconds

Ordered by: internal time
```

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
4231968	85.451	0.000	85.451	0.000	{method 'randint' of 'mtrand.RandomState' objects}
4231968	47.988	0.000	98.199	0.000	arraysetops.py:96(unique)
4231968	28.509	0.000	28.509	0.000	{built-in method numpy.core.multiarray.concatenate}
2000	18.198	0.009	202.838	0.101	<ipython-input-2-a6630299a345>:1(age_of_eve)
4231968	7.748	0.000	7.748	0.000	{method 'flatten' of 'numpy.ndarray' objects}
4231968	7.415	0.000	7.415	0.000	{method 'sort' of 'numpy.ndarray' objects}
4231968	5.214	0.000	6.539	0.000	numeric.py:484(asanyarray)
4231968	1.325	0.000	1.325	0.000	{built-in method numpy.core.multiarray.array}
4231968	0.990	0.000	0.990	0.000	{built-in method builtins.len}
1	0.020	0.020	202.858	202.858	<ipython-input-3-992538c1ebb7>:1(age_of_eve_trials)
1	0.000	0.000	202.858	202.858	{built-in method builtins.exec}
1	0.000	0.000	0.000	0.000	{built-in method numpy.core.multiarray.zeros}
1	0.000	0.000	202.858	202.858	<string>:1(<module>)
1	0.000	0.000	0.000	0.000	{method 'disable' of '_lsprof.Profiler' objects}

Hum, it seems that **randint** and **unique** are quite time consuming. Maybe looking into that direction allows to produce a faster code.

```
In [12]: cProfile.run('age_of_eve_trials_2()', sort = 'tottime')

4223849 function calls in 27.696 seconds

Ordered by: internal time
```

ncalls	tottime	percall	cumtime	percall	filename:lineno(function)
1	26.602	26.602	27.696	27.696	<ipython-input-11-c2a235d5435a>:1(age_of_eve_trials_2)
3332	0.687	0.000	0.687	0.000	{method 'randint' of 'mtrand.RandomState' objects}
4220492	0.407	0.000	0.407	0.000	{built-in method builtins.len}
21	0.000	0.000	0.000	0.000	{built-in method numpy.core.multiarray.zeros}
1	0.000	0.000	27.696	27.696	{built-in method builtins.exec}
1	0.000	0.000	27.696	27.696	<string>:1(<module>)
1	0.000	0.000	0.000	0.000	{method 'disable' of '_lsprof.Profiler' objects}

Well, **randint** and **unique** have been dealt with, and the code is now 6 times faster. It's your turn!

You may not gain that much, or maybe you'll gain a lot more than that. This does not matter: the goal is for you to develop a methodology for writing efficient code.

Answer:

2. The Voter Model

We now propose another approach inspired by game theory: the Voter model. We consider an undirected, connected graph $G = (V, E)$ with self-loops on all nodes. At time $t = 0$, each node has a given opinion (distinct nodes can share the same opinion, so there are between 1 and $n = |V|$ distinct opinions). At each time $t > 1$, each node picks up a neighbor at random and adopts its opinion at time $t - 1$. We want to study how the opinions evolve with time.

Question 1

Tell how the Voter model generalizes the Mitochondrial Eve model. Can you guess something about the evolution of opinions over time?

Answer:

Question 2

Let us do some math. Let P_t^a a vector of size n that tells for each node the probability that it has opinion a at time t : for each $i = 1, \dots, n$, the i -th component $P_t^a(i)$ of P_t^a is the probability that node i has opinion a at time t . In particular, the components of P_0^a are 1 over all nodes that have initially opinion a , 0 elsewhere.

2.1. Use conditional probabilities to express P_{t+1}^a as a function of P_t^a . Does it remind you of something? The answer may be yes or no depending on your background. If it is no, a course will be provided at this point.

2.2. Deduce the limit P_∞^a of P_t^a when $t \rightarrow +\infty$. To do this, you need to use the Perron-Frobenius Theorem recalled below. You can try to intuit the shape of the vector Q by solving the linear system $QA = Q$ for a small value of n (like $n = 3$) and then generalize your result for any value of n .

Hint: In case you have no background on Markov chains and stochastic process, you can admit the following result. Don't worry, we'll investigate this in more details later in the course and give some intuition of its meaning.

Right stochastic matrix Let $A = (a_{i,j})$ be a $n \times n$ matrix. We say that A is a *right* stochastic matrix if:

- The coefficients of A are non-negative: $\forall i, j, a_{i,j} \geq 0$.
- Each row sums to 1: $\forall i, \sum_j a_{i,j} = 1$.

Perron-Frobenius Theorem (a variant, actually) If A is right stochastic, irreducible and aperiodic (the last two terms will not be investigated here; just admit it is the case for the considered situation), then $A^t \xrightarrow[t \rightarrow \infty]{} B$, where B is the right stochastic matrix having all its rows equal to the same row vector Q defined as the unique normalized solution to the equation $QA = Q$.

Answer of 2.1.:

Answer of 2.2.:

Question 3

You are a high manager from the company Lodi (Lodi is a sort of Apple). You have the budget to select 1000 customers and offer them the brand new Lodi anvil, hoping that this helps you dominate the strategic market of smart anvils against your competitors. You bought from the social network company FriendFace their database, which tells you who's friend with whom. According to the Voter model, what should you do?

Answer:

Question 4 (bonus)

Verify your results with simulations (for instance with some Erdős-Rényi graph and 2 opinions).

Answer:

3. Epidemic Live Streaming

We complete our tour with a completely different framework: P2P Live streaming. There are n peers that want to watch a Live event. The Live stream is produced by a source that cuts it into chunks of 1 second and can deliver one chunk per second. Each peer has the capacity to deliver 1 chunk per second as well.