

IFT2015 :: autumn 2017

Duty 4: our ancestors

Introduction

This TP studies the genetic inheritance in a human population by simulation. In particular, we develop a simulation platform for random events (birth, coupling, death) in the lives of a population of *sims*, and we examine the coalescence of ancestral lineages.

The TP targets the following skills:

- implementation and use of priority queues
- use of symbol tables
- modeling and simulation of random events

A. The simulation

The simulation follows a population of virtual individuals, or *sims*, which are objects with the following attributes:

- parents: mother and father
- birth date
 - date of death
 - sex: man or woman

The life of each sim takes place through random events: birth, death and reproduction.

Random life pattern

Time is represented as a floating number on the $1.0 = 1$ year scale, starting at 0.0 with the founding population.

The **lifetime** of a sim is a random number, following the law of Gompertz-Makeham (https://fr.wikipedia.org/wiki/Modèle_de_Gompertz). This law combines a constant accident rate (of 1% per year by default), and an increasing mortality rate with age (doubling every 8 years by default).

The **reproduction** is described in the perspective of the mothers:

- mating is possible between two sims of different sex, and ages bounded by maximum and minimum (default, 16-73 for men, and 16-50 for women)
- the mother gives birth to children during her reproductive years, with the waiting time until mating following the exponential law with a fixed rate r (= Poisson process (https://fr.wikipedia.org/wiki/Processus_de_Poisson))
- when the time comes, the mother chooses a partner at random, with preference for her previous partner (10% probability to change partner by default)

Events

We track the evolution of the population by generating random events such as birth, death, or mating of sims. Each event applies to a specific sim, and during the time of its occurrence, it can generate other events to follow.

An event E is therefore an object with the attributes

- $E.subject$: subject of the event, a sim

- type of event: birth, death or mating
- `E.time`: time of the event, a non-negative floating number

The heart of the simulation is an event queue (`eventQ`) that removes the event as soon as possible: a min-tas ordered by time (`.time`).

```

1  void simulate(int n, double Tmax)
2  {
3      PQ eventQ = new PQ(); // file de priorité
4      for (int i=0; i<n; i++)
5      {
6          Sim fondateur = new Sim(); // sexe au hasard, naissance à 0.0
7          Event E = nouvel événement de naissance pour fondateur à 0.0
8          eventQ.insert(E); // insertion dans la file de priorité
9      }
10     while (!eventQ.isEmpty())
11     {
12         Event E = eventQ.deleteMin(); // prochain événement
13         if (E.time>Tmax) break; // arrêter à Tmax
14         if (E.subject.deathtime>E.time)
15         {
16             traiter événement E
17         }
18     }
19     ...

```

Event processing

Birth. At the birth of sim x at time t , we do the following:

- one draws a lifetime D at random - put on new death event for x , at time $t + D$.
- if x is a girl, then we take a waiting time A until reproduction - put on a new reproduction event for x , at time $t + A$.
- we record x in the population

Death. When sim x dies at time t , it is removed from the population.

Reproduction. A reproduction event of sim x (mother) in time t is processed by the following procedure:

- if x is dead, then nothing to do
- if x is of reproductive age, so choose a partner $there$ to have a baby with
 - create the baby sim with random sex, birth time t , and don the event of his birth
 - save x and y as the last partners to each other
- we draw a new waiting time A until reproduction - put on new reproduction event for x , at time $t + A$ (also if it is old or not)

Rules to choose the father. For a mother x already chosen, we select the father y at random, by a parameter f of "fidelity":

1. if x is in a relationship with z (previous baby with z who is still alive, and did not cheat with another woman):
 1. choose z with probability f
 2. or choose another suitable man (= living and reproductive age) uniformly, with probability $1-f$ (the solicited man always agrees here)
2. if x is not in a relationship (no previous baby, or dead / unfaithful partner), then she solicits candidates until acceptance: choose a suitable man $there$ uniformly at random - if not in a relationship he accepts without hesitation, or if he is in a relationship, he accepts with probability $1-f$.

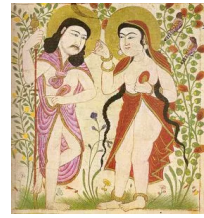
```

1  // Random RND = new Random();
2  Sim y = null;
3  if (!x.isInARelationShip || RND.nextDouble()>fidelite)
4  { // partenaire au hasard
5      do
6      {
7          z = random sim
8          if (z.getSex()!=x.getSex() && z.isMatingAge(E.time)) // isMatingAge() vérifie si z est de l'age
9          {
10             if (x.isInARelationShip() //
11                 || !z.isInARelationShip()
12                 || RND.nextDouble()>fidelite)
13                 { y = z; }
14             } while (y==null);
15     } else
16     {
17         y = partenaire précédent de x
18     }
19

```

B. Adam (s) and Eve (s): coalescence of lineages

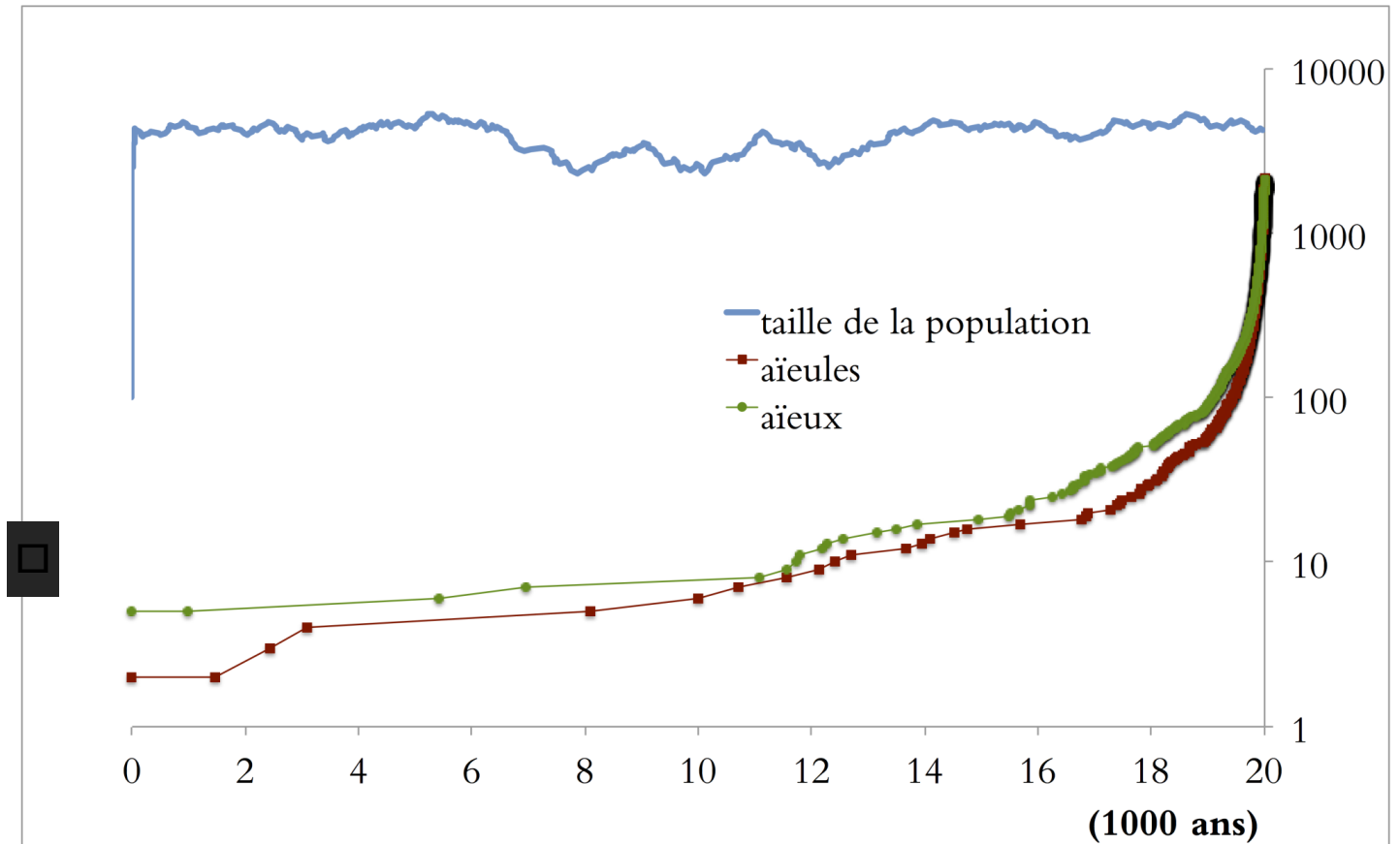
After simulating a few thousand years, we can trace the ancestors of the population of common sims. If we go back in time, the ancestral lines merge and we find less and less lines represented in the current population. We want to trace the number of ancestors to the current population as a function of time. For fathers (ancestors), we maintain a *PA* set of paternal alleles by retreating in time:



Adam and Eve in Manafi 'al-Hayawan

1. start with all men in the current population
2. repeat: remove the youngest individual (date of birth t) and add his father if he is not yet there. If the father is already there, then record the coalescence point (t, n), where the latter is the size of the *PA* set.

Continuing the iteration until only one father remains, or founders, we obtain all the points of coalescence, and the number of ancestral lineages at any time past. We do the same calculation for the mothers.



A population of ~ 5000 individuals for 20000 years (present time is on the right). Ancestral lineages (ancestors and ancestors) fuse rapidly for ~ 300 years, but more and more rarely in the more distant past.

C. Work to be done

Code for simulation.

Implement the code for the simulation described in A (sims, event, event queue). Model parameters : 3 Gompertz-Makeham parameters (see my code provided), 1 reproduction rate parameter r (adjusted to the mortality model to stabilize the population size), and 1 fidelity parameter f (= 90%).

- You'll find my code for generating random variables on github here: [AgeModel.java \(https://github.com/csurosm/IFT2015-A17/blob/master/src/pedigree/AgeModel.java\)](https://github.com/csurosm/IFT2015-A17/blob/master/src/pedigree/AgeModel.java). The method `randomAger` returns a random lifetime, and the method `randomWaitingTime(r)` gives a random wait time in a rate process r . The method `expectedParenthoodSpan(a,b)` calculates the number of years of expected reproduction between ages a and b (maternal age min and max) according to the mortality parameters. This is useful for choosing a stable reproduction rate: if $r = 2.0 / \text{expectedParenthoodSpan}$, then a woman gives birth to 2 babies on average during her life, and the population remains stable.
- You can also see my sketched implementation for sims: [Sim.java \(https://github.com/csurosm/IFT2015-A17/blob/master/src/pedigree/Sim.java\)](https://github.com/csurosm/IFT2015-A17/blob/master/src/pedigree/Sim.java).
- Note that it is necessary to have a mechanism to choose a sim randomly living for reproduction. I recommend a binary heap to store live sims, ordered by time of death. The sims can be easily removed when they die (eg, remove everyone with time of death before `E.time` when one deals with event `E`), and one can choose by the uniform index (`Random.nextInt(n)`) in the heap.

Code for coalescence

Implement the code to find the coalescence points (for fathers and mothers, separately) as described in B.

- In this task we must maintain the set of ancestors (*PA* above) with features (1) to remove the youngest and (2) check if an ancestor is already there. The easiest way is to work with a priority queue and a dictionary at the same time, but you can implement your own solution too.

Empirical study

Implement a software for execution at the command line with two arguments: *n* and *Tmax*, the size of the founder population and the maximum time for the simulation. This software should display three sets of data (on standard out):

1. population time and size (just sample every ~ 100 years)
2. time and number of coalesced paternal lineages
3. time and number of coalesced maternal lineages

Run your code with the same pair of *n* and *Tmax* of your choice ($n \geq 1000$, $T_{max} \geq 10 n$) 5-10 times, and observe the behavior of the coalescence.

modalities


Work in teams of 2.

Use freely available code on the Internet or other medium, but credit the source!

Submit in Studium (1) your code in a single JAR file, with the sources and classes compiled, and (2) a graph showing the result of one or more simulations.

Deadline for submission: November 30, 20:15.

Advertisements

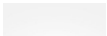


ARMANI EXCHANGE BLA...


the slim fit casual jean is garment dyed with a classic five pocket denim style. comfortable fit with...

\$45

Buy Now



ARMANI EXCHANGE NAV...




ARMANI EXCHANGE BLU...

this timeless straight-leg fit is updated with ample distressing and wear patterns, designed fo...

\$55

Buy Now



ARMANI EXCHANGE BLA...

☐ November 15, 2017

☐ csurosm

☐ duty

Create a free website or blog on WordPress.com.