

Projet 2. Nos ancêtres communs (10+1 point)

2.1 Introduction

Ce T.P. étudie la hérédité génétique dans une population humaine par simulation. En particulier, on développe une plate-forme de simulation d'événements aléatoires (naissance, couplage, mort) dans les vies d'une population de sims, et on examine la coalescence des lignées ancestrales. Le T.P. vise les compétences suivantes :


- ★ implémentation et usage de files de priorité
- ★ usage de tables de symboles
- ★ modélisation et simulation d'événements discrets aléatoires

Modalités

Le travail est conçu pour des équipes de deux : une des compétences importantes est la coordination de travail de programmation. Postez sur le Forum dans StudiUM pour former des équipes, et si vous acceptez un.e binôme au hasard. Ceux et celles qui sont d'accord, seront appariés au hasard. Ceci dit, il est possible de compléter le travail seul.e pour une pénalité modeste.

Évaluation

Soumettez un seul fichier JAR exécutable qui contient aussi les sources, et un rapport (en PDF) qui nomme les membres de l'équipe, décrit l'organisation de travail, explique votre implémentation, et montre les résultats de l'étude empirique de coalescence.

- (i) (1 point) organisation de travail d'équipe : documentez (a) l'organisation de la collaboration (p.e. rencontres, *pair programming*) ; (b) vos méthodes de communication et de gestion de code entre membres ; (c) la partition des tâches (l'équité sera prise en compte dans l'évaluation) ; et (d) les conflits (s'il y en a) et leurs résolution.
-  vous perdez cet 1 point si vous travaillez seul.e (car aucune coordination de travail) ; sauf si (a) vous avez une accommodation pour travail en équipe, ou (b) l'autre membre d'équipe a abandonné le travail, ou (c) la formation d'équipe était impossible (à la discrétion du prof, p.e. nombre impair d'étudiants à appairer). Déclarez votre raison dans la soumission.
- (ii) (9 points) correction, qualité et efficacité du code développé (entre autres, il faut code bien structuré et documenté dans la source, avec la séparation claire des responsabilités des classes, et l'utilisation des types abstraits et structures adéquates)
- (iii) (0 points mais un grand merci) SVP, rapportez combien de temps vous avez passé à travailler sur ce devoir : les heures de codage (écrire, tester, déboguer) et les heures de travail au total.

2.2 La simulation

La simulation suit une population d'individus virtuels, ou **sims**, qui sont des objets avec les attributs suivants :

- ★ parents : mère et père
- ★ date de naissance
- ★ date de décès
- ★ sexe : homme ou femme

La vie de chaque sim avance par des événements aléatoires : naissance, mort et reproduction. La simulation se déroule en temps en maintenant deux structures de données : la population de sims vivants, et la file d'événements.

Modèle de vie aléatoire

Le **temps** est représenté comme un nombre flottant sur l'échelle 1.0=1 an, débutant à 0.0 avec la population de fondateurs.

Le modèle assumé est un modèle réaliste de l'histoire des humaines¹. La **durée de vie** d'un sim est un nombre aléatoire, suivant la loi de Gompertz-Makeham². Cette loi combine un taux d'accident constant (de 1%/an par défaut), et un taux de mortalité croissant avec l'âge (se doublant chaque 8 an par défaut).

La **reproduction** est décrite dans la perspective des mères :

- (i) accouplement est possible entre deux sims de sexe différent, et d'âges bornés par minimum et maximum (par défaut, 16–65 pour homme, et 16–40 pour femme) ;
- (ii) la mère donne naissance à des enfants pendant ses années de reproduction, avec le temps d'attente jusqu'à accouplement suivant la loi exponentielle avec un taux fixe **r** : les événements forment un processus de Poisson³ ;
- (iii) quand le temps arrive, la mère choisit un partenaire au hasard, avec préférence pour son partenaire précédent (10% probabilité de changer partenaire par défaut)

Événements

On suit l'évolution de la population en générant des événements aléatoires tel que naissance, mort, ou accouplement de sims. Chaque événement applique à un sim spécifique, et lors du temps de son occurrence, il peut générer d'autres événements à suivre. Un événement *E* est donc un objet avec les attributs

- ★ **E.subject** : sujet de l'événement, un sim
- ★ type de l'événement : naissance, mort ou accouplement
- ★ **E.time** : temps de l'événement, un nombre flottant non-négatif

Le cœur de la simulation est une file d'événements (**eventQ**) qui permet de retirer l'événement le plus tôt : un min-tas ordonné par temps (**time**).

1. D. Rohde, S. Olson, and J. Chang. Modelling the recent common ancestry of all living humans. *Nature*, 431:562–566, 2004. DOI: 10.1038/nature02842
2. W₍₆₎:distribution de Gompertz-Makeham

3. W₍₆₎:processus de Poisson

```

void simulate(int n, double tMax){ // population et temps max
    PQ eventQ = new PQ(); // file de priorité
    for (int i=0; i<n; i++){
        Sim fondateur = new Sim(); // sexe au hasard, naissance à 0.0
        Event E = nouvel événement de naissance pour fondateur à 0.0
        eventQ.insert(E); // insertion dans la file de priorité
    }
    while (!eventQ.isEmpty()){
        Event E = eventQ.deleteMin(); // prochain événement par E.time
        if (tMax < E.time) break; // arrêter à tMax
        if (E.time < E.subject.deathTime){
            traiter événement E
        } // else rien à faire avec E car son sujet est mort
    }
    ...
}

```

Traitement d'événements

Naissance. Lors de la naissance de sim x à temps t , on fait le suivant :

- [n1] on tire une durée de vie D au hasard – enfiler nouvel événement de mort pour x , à temps $t + D$.
- [n2] si x est une fille, alors on tire un temps d'attente A jusqu'à reproduction – enfiler nouvel événement de reproduction pour x , à temps $t + A$.
- [n3] on enregistre x dans la population

Mort. Lors de la mort de sim x à temps t , on le retire de la population.

Reproduction. Un événement de reproduction de sim x (la mère) à temps t se traite par la démarche suivante :

- [r1] si x est morte, alors rien à faire
- [r2] si x est d'âge de reproduction (sinon, juste passer à **r3**), alors choisir un partenaire y pour avoir un bébé avec
 - ★ créer le sim bébé avec sexe au hasard, temps de naissance t , et enfiler l'événement de sa naissance
 - ★ enregistrer x et y comme derniers partenaires l'une à l'autre
- [r3] on tire un nouveau temps d'attente A jusqu'à reproduction – enfiler nouvel événement de reproduction pour x , à temps $t + A$ (également si elle est d'âge ou non)

Paternité. Pour une mère x déjà choisie, on sélectionne le père y au hasard, par un paramètre de fidélité f (par défaut, 90%) :

- [p1] si x est dans une relation avec z (bébé précédent avec z qui est toujours vivant, et n'a pas triché avec une autre femme) :
 - [p1.1] choisir z avec probabilité f
 - [p1.2] ou choisir un autre homme adéquat (=vivant et d'âge de reproduction) uniformément, avec probabilité $(1 - f)$: l'homme sollicité accepte toujours l'offre, même s'il a un partenaire précédent
- [p2] si x n'est pas dans une relation (aucun bébé précédent, ou x a un partenaire mort/infidèle), alors elle sollicite des candidats jusqu'à acceptation : choisir un homme adéquat y uniformément au hasard – s'il n'est pas dans une relation, il accepte sans hésitation, ou s'il est dans une relation, il accepte avec probabilité $(1 - f)$.

```

// Random RND = new Random(); // générateur de nombres aléatoires
Sim y = null; // choisir pere y
if (!x.isInARelationShip || RND.nextDouble()>fidelite)
{ // partenaire au hasard
    do
    {
        z = random sim
        if (z.getSex()!=x.getSex() && z.isMatingAge(E.time))
            // isMatingAge() vérifie si z est de l'age acceptable
        {
            if (x.isInARelationShip() // z accepte si x est infidèle
                || !z.isInARelationShip()
                || RND.nextDouble()>fidelite)
            { y = z;}
        }
    } while (y==null);
} else
{
    y = partenaire précédent de x
}

```

2.3 Adam(s) et Ève(s) : coalescence de lignées

Après avoir fait la simulation de quelques milliers d'années, on peut retracer les ancêtres de la population de sims courants. Si on remonte le temps à l'inverse, les lignées ancestrales se fusionnent et on trouvera moins et moins lignées représentées dans la population courante. On veut tracer le nombre d'ancêtres à la population courante en fonction de temps. Pour les pères (aïeux), on maintient un ensemble **PA** d'allèles paternels en reculant dans le temps :

1. commencer avec tous les hommes dans la population courante, enregistrer (p, n) où p est temps présent et n est la taille de **PA**
2. répéter : retirer l'individu le plus jeune (date de naissance t) de **PA** et ajouter son père s'il n'est pas encore là. Si le père est déjà là, alors on a coalescence : mettre $n \leftarrow n - 1$ et enregistrer (t, n) , où ce dernier est la nouvelle taille de l'ensemble **PA**

En continuant l'itération jusqu'à ce qu'il ne reste qu'un seul père, ou que des fondateurs, on obtient tous les points de coalescence, et le nombre de lignées ancestrales à tout temps passé. On fait le même calcul pour les mères. Si t_{Max} est assez grand, on finit avec le temps de coalescence complète $(t_0, 1)$; sinon avec $(0, n_0)$ le nombre de fondateurs n_0 représentés dans la population courante.

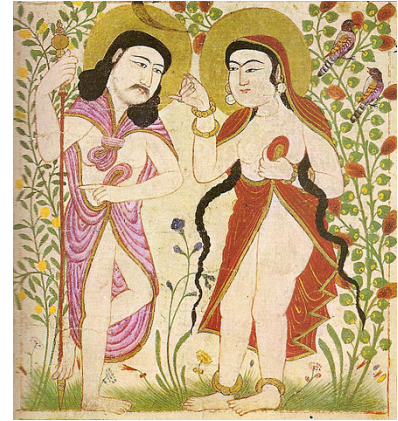
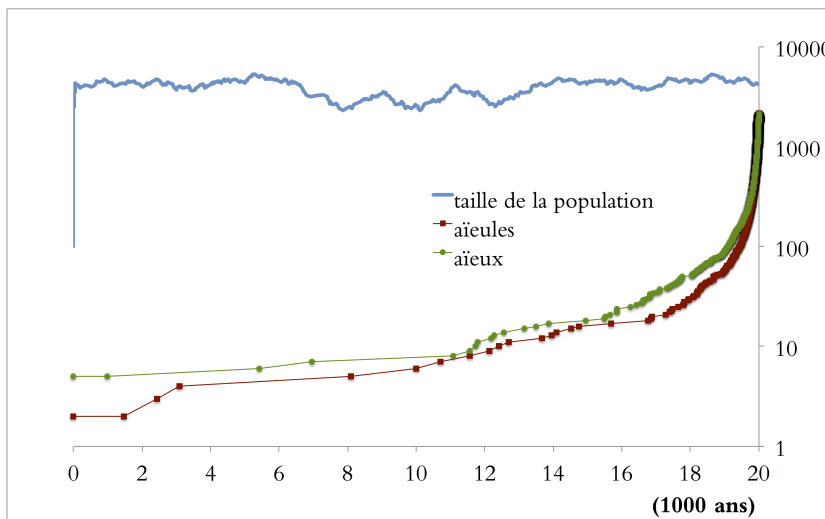


FIGURE 1: Adam et Ève dans le *Manafi al-Hayawan*

FIGURE 2: Une population de $5000 \pm$ individus pendant 20000 ans (temps présent à la droite). Les lignées ancestrales (aïeules et aïeux) se fusionnent rapidement pendant les $300 \pm$ récents mais de plus en plus rarement dans le passé plus distant.

2.4 Travail à faire

Code pour simulation

► Implémenter le code pour la simulation décrite en §2.2 (sims, événement, file d'événements). Paramètres du modèle : 3 paramètres de Gompertz-Makeham (v. mon code fourni), 1 paramètre de taux de reproduction r (ajusté au modèle de mortalité pour stabiliser la taille de la population), et 1 paramètre de fidélité f (=90% par défaut).

- ★ Vous trouvez mon code pour la génération de variables aléatoires dans StudiUM : `AgeModel.java`. La méthode `randomAge` retourne une durée de vie au hasard, et la méthode `randomWaitingTime(r)` donne un temps d'attente aléatoire dans un processus de taux r . La méthode `expectedParenthoodSpan(a, b)` calcule le nombre d'années de reproduction en espérance entre ages a et b (age de maternité min et max) selon les paramètres de mortalité. C'est utile pour choisir un taux de reproduction stable : si $r = 2.0/\text{expectedParenthoodSpan}$, alors une femme donne naissance à 2 bébés en moyenne pendant sa vie, et la population reste stable.
- ★ Vous pouvez également consulter mon implémentation esquissée pour les sims : `Sim.java`.
- ★ Notez qu'il est nécessaire d'avoir un mécanisme de choisir un sim vivant au hasard pour la reproduction. Utilisez un tas binaire (ou d -aire) pour stocker les sims vivants, ordonnés par temps de mort. On peut retirer facilement les sims quand ils meurent (p.e., enlever tout le monde avec temps de mort avant $E.\text{time}$ quand on traite événement E), et on peut en choisir par l'indice uniforme (`Random.nextInt(n)`) dans le tas.

Code pour coalescence

► Implémenter le code pour retrouver les points de coalescence (pour pères et mères, séparément) comme décrit en §2.3.

Dans cette tâche on doit maintenir l'ensemble d'ancêtres (PA ci-haut) avec des fonctionnalités (1) de retirer le plus jeune, et (2) de vérifier si un ancêtre est déjà là. Le plus simple est de travailler avec une file de priorité et un dictionnaire (`java.util.Map`) ou ensemble (`java.util.Set`) en même temps.

Structure de tas. Pour ce TP, il est nécessaire d'écrire sa propre implémentation de tas binaire (ou tas d -aire avec $d = 4$ ou $d = 8$ p.e). Il est une bonne idée d'écrire une classe générique, instanciée avec un `Comparator`, et alors l'utiliser également pour suivre la population de sims vivants en avançant ou en reculant dans le temps, ou pour stocker la chaîne d'événements à traiter.

Étude empirique

► Implémenter un logiciel pour exécution à la ligne de commande avec deux arguments : n et $tMax$, la taille de la population fondateur et le temps maximal pour la simulation. Ce logiciel devrait afficher trois séries de données (sur la sortie standard) — voir l’illustration sur Figure 2 :

- ★ temps et taille de population (il suffit d’échantillonner chaque 100 ans) ;
- ★ temps et nombre de lignées paternelles coalescées ;
- ★ temps et nombre de lignées maternelles coalescées.

Exécutez votre code avec le même paire de n et $tMax$ de votre choix ($1000 \leq n; 10n \leq tMax$) 5—10 fois (ou plus, si vous voulez statistiques), et observez le comportement de la coalescence.

►► (+1 point) Pour points boni, analysez la distribution de coalescence (jusqu’à 1 lignée : «Adam» pour pères et «Ève» pour mères) à travers au moins quelques douzaines de simulations, et (1) déterminez si la différence d’âge min-max entre femmes et hommes mène à Adam et Ève dans des temps différents (si coalescence et plus ou moins rapide) ; **ou** (2) postulez un hypothèse sur la relation entre un paramètre choisi (le plus intéressant est la taille de la population n) et le temps de coalescence complète (inspectez la moyenne et l’écart-type dans les expériences répétés avec paramètres identiques).

Références

D. Rohde, S. Olson, and J. Chang. Modelling the recent common ancestry of all living humans. *Nature*, 431 :562–566, 2004. DOI: 10.1038/nature02842.