Université UCA

Rapport de Projet

Lab # 4 - Lab 4 Modèle Toy case SEIR multi-agent

Fleury Guillaume & Veysseyre Thomas

4 avril 2024

Lab # 4 - Lab 4 Modèle Toy case SEIR multi-agent

Fleury Guillaume & Veysseyre Thomas

4 avril 2024

Table des matières

Table des figures

1 Introduction

Ce rapport détaille la création d'un modèle multi-agent SEIR pour simuler la diffusion d'une maladie dans une population structurée sur une grille toroïdale. Le modèle simule les interactions entre agents susceptibles, exposés, infectés et récupérés, en intégrant des comportements stochastiques tels que le mouvement aléatoire et la transmission de l'infection. Les analyses des dynamiques épidémiques issues de ces simulations sont réalisées via Jupyter Notebooks, permettant une visualisation et une interprétation des tendances à travers des données générées et des graphiques pertinents.

2 Méthodologie

Dans le cadre de ce projet, nous avons conçu et implémenté un modèle SEIR multi-agent afin d'étudier la propagation d'une infection au sein d'une population théorique. Ce modèle repose sur une approche stochastique pour simuler le mouvement et l'interaction des agents sur une grille de simulation. La grille elle-même est conçue comme une surface toroïdale pour éviter les artefacts liés aux bords dans la dynamique de la simulation.

2.1 Description du Modèle SEIR Multi-Agent

Notre modèle SEIR multi-agent catégorise les individus en quatre états distincts : Susceptible (S), Exposé (E), Infecté (I), et Récupéré (R), chacun correspondant à une phase différente dans la progression de la maladie. Les agents susceptibles sont initialement sains mais peuvent être exposés au virus. Une fois exposés, ils entrent dans une phase latente pendant laquelle ils ne sont pas contagieux. Après une période définie, ces individus deviennent infectieux. Finalement, les individus infectés peuvent se rétablir et acquérir une immunité temporaire ou permanente.

2.2 Choix du PRNG

Le Mersenne Twister est choisi comme générateur de nombres pseudo-aléatoires (PRNG) pour notre simulation en raison de sa longue période et de sa distribution uniforme. Ces caractéristiques sont essentielles pour simuler de manière réaliste le comportement aléatoire dans des processus tels que le mouvement des agents et leurs transitions d'état de santé.

Utilisation : Employé à travers la simulation, le Mersenne Twister assure une variabilité naturelle dans les interactions entre agents et dans l'évolution de la maladie. Par exemple, il détermine aléatoirement si un agent susceptible devient exposé en fonction de la proximité d'agents infectés :

```
if ("S".equals(agent.getStatus()) && random.nextDouble() < infectionChance) {
    agent.setStatus("E");
}
```

Avantages : La performance et la fiabilité du Mersenne Twister garantissent que nos simulations produisent des résultats précis et reproductibles, cruciaux pour l'analyse des dynamiques épidémiologiques.

Le choix de ce PRNG est donc stratégique pour atteindre une modélisation fidèle des phénomènes stochastiques dans notre étude des épidémies.

2.3 Processus de Simulation

La simulation évolue en cycles temporels, représentant par exemple des jours. À chaque cycle, les agents peuvent se déplacer et changer d'état en fonction de règles probabilistes. Le mouvement des agents est déterminé par une sélection aléatoire de nouvelles coordonnées sur la grille, illustrée par le code suivant :

```
int newX = ran.nextInt(mapSize);
int newY = ran.nextInt(mapSize);
buddy.changePos(newX, newY);
```

La propagation de l'infection est simulée en calculant la probabilité d'exposition basée sur le nombre de voisins infectés. Si un agent susceptible a un ou plusieurs voisins infectés, il peut devenir exposé avec une probabilité qui augmente avec le nombre de voisins infectés. La mise à jour de l'état de santé se fait ensuite conformément aux durées caractéristiques associées à chaque état :

```
if (status.equals("E") && statusTime > dE) {
    status = "I";
    statusTime = 0;
} else if (status.equals("I") && statusTime > dI) {
    status = "R";
    statusTime = 0;
} else if (status.equals("R") && statusTime > dR) {
    status = "S";
```

```
9       statusTime = 0;
10  }
```

Chaque état a une durée définie par une distribution exponentielle négative, assurant une modélisation plus réaliste du temps passé dans chaque phase de la maladie. Les résultats de la simulation, stockés sous forme de données temporelles, sont ensuite utilisés pour générer des graphiques qui permettent une analyse quantitative des modèles de propagation et l'efficacité des interventions simulées.

3 Implémentation

L'architecture de notre simulation de propagation d'une maladie infectieuse au sein d'une population repose sur un modèle SEIR multi-agent, implémenté en Java. La structure du code est conçue pour modéliser les interactions complexes entre les individus de la population et pour suivre l'évolution de l'état de santé de chaque individu au cours du temps.

3.1 Structure du Code

La logique de notre simulation est divisée en trois classes principales qui représentent les différents aspects du système : 'execute', 'human', et 'map'.

- La classe **'execute'**, le cœur de la simulation, orchestre le déroulement des événements, de l'initialisation de la grille et des agents jusqu'à la collecte et l'enregistrement des résultats de la simulation. Elle initialise les agents avec des états de santé spécifiques et gère les cycles de simulation, où chaque agent peut se déplacer et changer d'état selon des règles prédéfinies.
- La classe **'human'** définit les attributs et les comportements des agents. Chaque agent a un état de santé (Susceptible, Exposé, Infecté, Récupéré), ainsi que des durées associées à chaque état. Les agents ont également la capacité de se déplacer sur la grille, simulant ainsi les interactions sociales et la propagation de la maladie.
- La classe **'map'** représente l'environnement dans lequel les agents interagissent. Cette grille toroïdale permet aux agents de se déplacer d'un bord à l'autre, créant un espace continu sans frontières artificielles. Elle stocke la position de chaque agent et facilite la vérification des voisins pour la transmission de la maladie.

La structure modulaire de notre code facilite la compréhension des mécanismes de simulation et permet une adaptation flexible aux différents scénarios épidémiologiques. Les interactions entre les classes sont soigneusement gérées pour refléter les dynamiques réalistes de propagation des maladies au sein d'une population.

3.1.1 Détailles du Code Source

Notre implémentation fait appel à des pratiques de développement avancées pour assurer la cohérence et l'efficacité de la simulation. Au cœur du code source, des mécanismes de contrôle des flux permettent une exécution fluide et des transitions d'état logiques pour chaque agent.

Gestion des États et Transitions. Les états de santé des individus sont gérés par un ensemble de transitions d'état au sein de la classe **'human'**. Ces transitions sont déclenchées par des événements spécifiques simulés dans le temps discret de notre modèle. La flexibilité de ces transitions est telle qu'elle permet d'intégrer facilement des modifications des paramètres épidémiologiques ou des ajouts de nouveaux états de santé, le cas échéant.

Modularité et Réutilisation du Code. Les classes ont été conçues pour être aussi modulaires que possible, permettant ainsi la réutilisation dans différents contextes de simulation. Par exemple, la classe **'map'** peut être étendue pour inclure des caractéristiques géographiques ou des obstacles, tandis que la classe **'human'** peut être enrichie pour inclure des stratégies de déplacement plus sophistiquées.

Optimisation et Performance. Des considérations importantes ont été prises pour optimiser la performance de la simulation, en particulier dans la gestion des collections d'agents et la mise à jour de leur état dans la grille. L'efficacité des structures de données utilisées minimise l'empreinte mémoire et les coûts de calcul, essentiels pour la simulation de grandes populations sur de nombreux cycles.

Perspectives d'Évolution. Le cadre actuel de la simulation a été préparé en gardant à l'esprit les futures évolutions. Des points d'extension ont été identifiés, permettant l'intégration de nouvelles fonctionnalités, telles que des interventions de santé publique ou des modèles de déplacement variables basés sur le comportement des individus ou des politiques extérieures.

En définitive, notre code source est un reflet de notre engagement envers des simulations de haute fidélité. Il est le résultat d'une planification minutieuse et d'une compréhension approfondie de la dynamique de la propagation des maladies infectieuses.

3.1.2 Création des Agents

Les agents, représentant les individus au sein de la simulation, sont instanciés par la classe **'human'**. Chaque agent est créé avec un état de santé initial qui peut être Susceptible, Exposé, Infecté ou Récupéré. Cet état définit le comportement de l'agent dans le contexte de la simulation, notamment sa capacité à se déplacer et à interagir avec les autres agents.

La création des agents s'effectue dans le constructeur de la classe **'execute'**, où un ensemble d'agents est initialisé avec des états aléatoires, basés sur une distribution définie pour refléter les conditions initiales de la simulation. Les agents sont ensuite placés sur la grille, gérée par la classe **'map'**, à des positions aléatoires pour commencer la simulation.

Le code suivant illustre la création d'un agent avec un état initial et l'utilisation du générateur de nombres pseudo-aléatoires pour définir la durée de chaque état de santé selon une distribution exponentielle négative :

```
public human(String status, MersenneTwister ran) {
    this.status = status;
    this.dE = (int)ran.negExp(3);
    this.dI = (int)ran.negExp(7);
    this.dR = (int)ran.negExp(365);
    this.statusTime = 0;
}
```

Cette méthode garantit que chaque agent possède des attributs uniques dès le départ, ce qui contribue à la diversité des interactions et des trajectoires de maladie au sein de la population modélisée. La création des agents avec ces caractéristiques permet de simuler plus fidèlement la propagation de la maladie dans des conditions variées.

3.1.3 Mise à Jour des États

La dynamique du modèle SEIR est capturée par les transitions entre les différents états de santé des agents : de Susceptible à Exposé, d'Exposé à Infecté, puis d'Infecté à Récupéré. Ces transitions sont modélisées par la méthode `update()` de la classe **'human'**, qui est invoquée à chaque cycle de simulation pour chaque agent.

Au début de chaque cycle, la grille est parcourue pour identifier les agents exposés et infectés. Les agents exposés deviennent infectés après une période déterminée par leur attribut `dE`, simulant la période d'incubation de la maladie. De même, les agents infectés passent à l'état de récupérés après la durée spécifiée par `dI`, représentant la période pendant laquelle ils sont contagieux. Les agents récupérés retournent à l'état susceptible après un intervalle `dR`, reflétant une possible perte d'immunité.

La logique de mise à jour est illustrée ci-dessous :

```
public void update() {

    this.statusTime += 1;


    if ("E".equals(this.status) && this.statusTime > this.dE) {
        this.status = "I";
        this.statusTime = 0;
    } else if ("I".equals(this.status) && this.statusTime > this.dI) {
        this.status = "R";
        this.statusTime = 0;
    } else if ("R".equals(this.status) && this.statusTime > this.dR) {
        this.status = "S";
        this.statusTime = 0;
    }
}
```

Ce processus assure que la simulation reflète fidèlement les phases naturelles de la maladie, depuis l'exposition jusqu'à la guérison ou le retour à la susceptibilité. La mise à jour des états des agents permet de simuler l'évolution de l'épidémie sur la grille, influençant ainsi les interactions futures entre les agents et la propagation globale de la maladie dans la population simulée.

3.1.4 Autres Fonctionnalités Importantes

En plus de la création des agents et de la mise à jour de leurs états, notre simulation intègre plusieurs autres fonctionnalités clés pour une modélisation réaliste et approfondie de la propagation des maladies.

Environnement de Simulation : La classe **'map'** crée une grille toroïdale qui sert d'environnement pour les agents. Cet environnement permet une mobilité continue des agents sans les restrictions imposées par les bords, imitant ainsi un espace clos où la maladie peut se propager sans obstacle géographique.

```
public void addHuman(int x, int y, human buddy) {
    grid[x][y].add(buddy);
    buddy.changePos(x, y);
}
```

Gestion des Interactions : Les interactions entre agents sont au cœur de la dynamique de transmission de la maladie. Lors de chaque cycle de simulation, la grille est parcourue pour déterminer les agents susceptibles d'être exposés en fonction de leur proximité avec des agents infectés. Cette logique d'interaction est cruciale pour simuler la propagation de l'infection.

```
private int infectedNeighbours(human buddy){
    int nbInfected = 0;
    return nbInfected;
}
```

Collecte et Analyse des Données : À chaque étape de la simulation, des données sont collectées concernant l'état de santé des agents, leur distribution sur la grille, et la progression générale de l'épidémie. Ces données sont ensuite utilisées pour générer des visualisations et des analyses statistiques, permettant d'évaluer l'efficacité des interventions simulées et de comprendre les mécanismes sous-jacents de la propagation de la maladie.

```
public String debugGlobalStatus(){
    return s + "," + e + "," + i + "," + r;
}
```

Ces fonctionnalités enrichissent la simulation en fournissant un cadre pour l'étude détaillée des épidémies et en offrant des outils pour l'analyse des résultats, contribuant ainsi à une meilleure compréhension des facteurs influençant la propagation des maladies infectieuses.

```
public static void main(String[] args) {
    execute exe = new execute(mapSize, nbHumanSus, nbHumanIll);
    for (int i = 0; i < nbIteration; i++) {
        exe.map.update(exe.ran);
        updateLoadingBar(i, nbIteration);
        results.add(exe.map.debugGlobalStatus());
    }
    writeStringsToCSV("results/result.csv", results.toArray(new String[0]));
}
```

Ces composants forment le cœur de notre système de simulation, permettant une exploration détaillée de la dynamique de propagation des maladies dans une population modélisée.

4 Résultats et Analyses

Dans le cadre de notre étude sur la dynamique de la propagation d'une maladie au sein d'une population, nous avons procédé à une série d'analyses statistiques en utilisant les données issues de notre modèle SEIR multi-agent. Ces analyses ont été réalisées en exploitant des fichiers CSV générés par la simulation et traités à l'aide de scripts Python.

4.1 Analyse des Données

Les fichiers CSV contenant les états quotidiens des agents ont été importés dans un environnement Python où des statistiques descriptives ont été calculées, incluant les valeurs moyennes, maximales, minimales et l'écart type pour chaque état de santé des agents : Susceptible, Exposé, Infecté, et Récupéré. Les résultats de ces analyses sont résumés dans les figures présente dans L'annexe (2)(3).

4.2 Évolution Temporelle des États

L'évolution de l'épidémie à travers le temps a été visualisée en traçant le nombre moyen d'agents dans chaque état de santé au cours de nos simulations. Les courbes temporelles moyennes mettent en évidence la séquence typique de l'épidémie : une montée rapide du nombre d'agents exposés et infectés suivie d'une phase de récupération où le nombre d'agents dans l'état de récupéré augmente. Voir annexe (4)(5)

4.3 Consultation des Analyses Détaillées

Pour une exploration approfondie de ces résultats, les analyses détaillées, y compris des statistiques supplémentaires et des observations spécifiques, sont disponibles dans le notebook Jupyter accompagnant ce projet. Le notebook fournit un compte-rendu exhaustif de la méthodologie d'analyse, des visualisations de données supplémentaires, et une interprétation des tendances observées. Il est disponible pour consultation en tant que fichier PDF joint ou peut être exécuté pour une interaction en direct avec les données.

Nous encourageons les lecteurs à se référer au notebook Jupyter pour une compréhension complète des dynamiques sous-jacentes à notre simulation et des conclusions dérivées de cette étude. Pour accéder à ces analyses, veuillez naviguer dans le dossier des résultats et ouvrir les fichiers correspondants.

— Notebook Jupyter : `Rapport/Etudes_Stats.ipynb`
— Document PDF : `Rapport/Etudes_Stats.pdf`

5 Discussion

Cette section aborde l'interprétation des résultats obtenus, les limitations identifiées dans notre étude et les suggestions d'améliorations pour les travaux futurs.

5.1 Interprétation des Résultats

L'analyse de nos simulations du modèle SEIR a révélé des dynamiques de propagation de la maladie qui sont cohérentes avec les modèles épidémiologiques théoriques. La baisse rapide du nombre d'agents susceptibles suivie par un pic dans les états exposés et infectés montre une transmission rapide de la maladie. Par ailleurs, l'augmentation constante et significative du nombre d'agents récupérés démontre l'efficacité des processus de guérison ou de l'acquisition d'immunité au sein de la population modélisée. Cependant, les variations notables entre les simulations individuelles suggèrent que de nombreux facteurs, y compris le hasard, peuvent influencer l'issue d'une épidémie.

5.2 Limitations

Notre étude comporte plusieurs limitations qui doivent être prises en compte lors de l'interprétation des résultats. Premièrement, la modélisation repose sur des hypothèses simplifiées concernant les comportements humains et les mécanismes de transmission de la maladie. De plus, notre modèle ne tient pas compte de facteurs environnementaux ou socio-économiques qui peuvent affecter la propagation. Deuxièmement, bien que le modèle SEIR soit robuste pour comprendre les tendances générales, il ne peut pas prédire avec précision les résultats pour des individus spécifiques ou des scénarios réels complexes.

5.3 Suggestions d'Améliorations

Pour améliorer les futures simulations et analyses, nous recommandons plusieurs axes de développement. L'incorporation de paramètres adaptatifs qui peuvent changer en réponse à des interventions de santé publique, comme la vaccination ou la mise en quarantaine, pourrait offrir une meilleure représentation de la réalité. Intégrer des modèles de mobilité plus complexes et des interactions basées sur des réseaux sociaux permettrait également d'ajouter de la profondeur à la simulation. Enfin, appliquer des techniques d'analyse plus avancées, telles que l'apprentissage automatique pour l'optimisation des paramètres, pourrait affiner la précision du modèle.

Ces améliorations pourraient aider à créer des modèles de simulation plus réalistes et instructifs, capables de guider efficacement la prise de décision en matière de santé publique.

6 Conclusion

6.1 Récapitulatif des Découvertes

Au terme de notre étude, nous avons dégagé des tendances significatives et des comportements caractéristiques de la propagation d'une maladie au sein d'une population simulée. Le modèle SEIR multi-agent a permis de capturer les étapes clés d'une épidémie, reflétant la transition des individus à travers les états de susceptibilité, d'exposition, d'infection et de récupération. Les simulations ont révélé une variabilité dans les trajectoires de la maladie, suggérant l'influence de multiples facteurs dynamiques dans l'évolution d'une épidémie.

6.2 Implications et Conclusions

Les résultats obtenus à partir de notre modèle fournissent des aperçus précieux sur la dynamique des maladies infectieuses, soulignant l'importance des mesures préventives et de contrôle pour en atténuer l'impact. Ces découvertes renforcent la valeur des modèles multi-agents en tant qu'outils pour la recherche épidémiologique et pour l'élaboration de stratégies de santé publique. Tout en reconnaissant les limitations inhérentes à toute modélisation, les implications de cette recherche soulèvent des considérations pertinentes pour la planification des interventions en cas d'épidémie et pour la préparation des systèmes de santé à des événements similaires à l'avenir.

7 Annexe

```java
public static void main(String[] args) throws Exception {
    // Initialization of simulation parameters.
    int mapSize = 300;
    int nbHumanSus = 19990;
    int nbHumanIll = 10;
    int nbIteration = 730;
    MersenneTwister ran = new MersenneTwister(4357);
    int nbExperiments = 100;
```

FIGURE 1 – Initialisation des paramètres de l'expérience, incluant la taille de la carte, le nombre d'humains sains et infectés, le nombre d'itérations et le nombre d'expériences.

```
----- Statistics for Individual Runs -----
Mean values for each state:
Susceptible     2183.281699
Exposed          192.954082
Infected         956.253260
Recovered      16667.510959
dtype: float64

Maximum values for each state:
Susceptible    19990
Exposed         4400
Infected       13321
Recovered      18808
dtype: int64

Minimum values for each state:
Susceptible    46
Exposed         0
Infected        6
Recovered       0
dtype: int64

Standard deviation for each state:
Susceptible    2287.602710
Exposed         275.545258
...
Recovered      2586.719966
dtype: float64
----------------------------------------
```

FIGURE 2 – Statistiques descriptives pour les états individuels.

```
----- Statistics for Average Curve -----
Mean values for each state:
Susceptible     2183.281699
Exposed          192.954082
Infected         956.253260
Recovered      16667.510959
dtype: float64

Maximum values for each state:
Susceptible    19990.00
Exposed         4074.66
Infected       12993.23
Recovered      18734.70
dtype: float64

Minimum values for each state:
Susceptible    65.7
Exposed         0.0
Infected        8.8
Recovered       0.0
dtype: float64

Standard deviation for each state:
Susceptible    2284.197513
Exposed         270.984166
...
Recovered      2586.476031
dtype: float64
----------------------------------------
```

FIGURE 3 – Courbe moyenne des états SEIR au cours du temps.

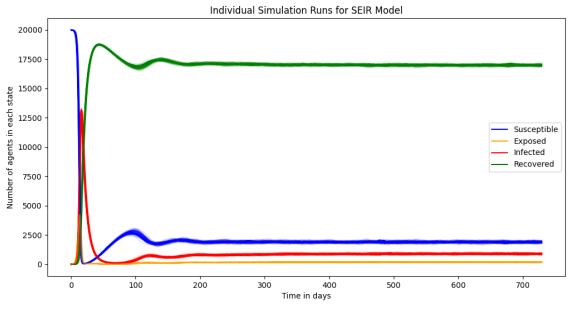

FIGURE 4 – Simulations individuelles pour le modèle SEIR.

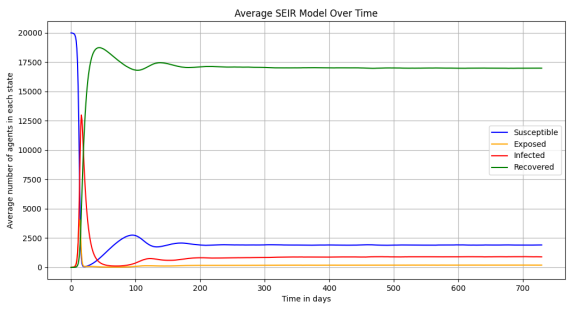

FIGURE 5 – Moyenne du modèle SEIR sur le temps.