



*Bachelor Universitaire de Technologie Informatique*

## RAPPORT DE STAGE

---

# SIMULATION MULTI-AGENTS DE STRATÉGIES DE GESTION DE LA PROPAGATION DES ÉPIDÉMIES

---



UNIVERSITATEA  
DIN  
CRAIOVA

Étudiant : Mohamed MESRI

Maître de Stage : Costin BĂDICĂ

Professeur Référent : Maria Cristina ONETE

08 avril - 15 juin 2024



## Remerciements

Je souhaite exprimer ma gratitude envers mon maître de stage, Monsieur Costin Bădică professeur à l'Université de Craiova, pour son accueil chaleureux, ses encouragements et l'opportunité qu'il m'a offerte de réaliser ce projet.

Je tiens à remercier Madame Maria Cristina Onete, ma tutrice de stage et enseignante à l'Institut Universitaire de Technologie du Limousin, pour son soutien constant tout au long de mon stage, sa disponibilité lorsque j'en ai eu besoin et pour m'avoir permis d'effectuer ce stage en Roumanie.

Mes remerciements vont également à Monsieur Laurent Bourdier, professeur à l'Institut Universitaire de Technologie du Limousin, pour m'avoir permis d'effectuer ce stage en Roumanie.

Je souhaiterais adresser mes remerciements à Ludivine, responsable des relations internationales à l'Institut Universitaire de Technologie du Limousin, pour son assistance précieuse dans les démarches administratives liées au programme Erasmus.

Enfin, je tiens à exprimer ma reconnaissance envers Madame Gabriela Anca Mic, responsable des relations internationales à l'Université de Craiova, pour son aide précieuse dans la recherche d'un logement à Craiova.

## Table des matières

Introduction .....	6
1 Contexte du projet.....	7
1.1 L'université de Craiova .....	7
1.1.1 Historique.....	7
1.1.2 Groupe de recherche.....	7
1.1.3 Recherches sur la modélisation des maladies infectieuses .....	8
1.2 Expression des besoins .....	8
1.2.1 Notions importantes .....	8
1.2.2 Compréhension du modèle multi-agent SIR et création d'un ABMS .....	9
1.2.3 Ajout de nouvelles stratégies contre une épidémie .....	9
1.2.4 Application web .....	10
1.3 Organisation et outil.....	10
1.3.1 Environnement de travail.....	10
1.3.2 Outils techniques.....	11
2 Mise en place du premier ABMS .....	12
2.1 Le travail de recherche existant .....	12
2.1.1 Le modèle spatial SIR de base .....	12
2.1.2 Modélisation de la propagation de l'infection .....	13
2.1.3 Gestion de l'infection .....	14
2.1.4 Représentation multi-agent .....	15
2.2 Développement .....	16
2.2.1 Structure du Framework MESA .....	16
2.3 Résultats .....	22
3 Ajout de nouvelles stratégies .....	24
3.1 Système de création de zones statiques .....	24
3.1.1 Développement .....	24
3.1.2 Résultat.....	25
3.2 Système de création de zones dynamiques .....	26
3.2.1 <i>K – Means</i> .....	27
3.2.2 <i>K – Means + +</i> .....	29
3.2.3 Score silhouette.....	31
3.2.4 Développement .....	32

3.2.5	Résultat .....	34
3.3	Système de port du masque .....	35
3.3.1	Port du masque sur toute la population.....	37
3.3.2	Port du masque dans les zones d'épidémies .....	39
3.3.3	Port du masque dans des zones d'épidémies et mobilité réduite .....	40
3.4	Bilan.....	41
4	Application Web.....	42
4.1	Interface utilisateur.....	42
4.2	Eléments de comparaisons .....	44
4.3	Bilan.....	45
	Conclusion .....	46
	Glossaire .....	47
	Bibliographie.....	50
	Annexes .....	51

## Introduction

Le stage représente une étape cruciale dans la vie d'un étudiant, car il vise à nous professionnaliser en mettant en pratique les compétences acquises au cours de nos années d'études. L'Institut Universitaire de Technologie du Limousin propose la possibilité d'effectuer un stage à l'étranger, soumis à l'examen des dossiers et à une sélection rigoureuse. J'ai saisi cette opportunité et ai été retenu pour réaliser mon stage du 06 avril au 15 juin 2024 en Roumanie, plus précisément à Craiova, où j'ai mené un projet de recherche sur les systèmes multi-agents de propagation d'épidémies au sein de leur université.

Mon projet s'est déployé selon trois axes majeurs : la compréhension du modèle spatial multi-agent SIR élaboré par M. Bădică et sa mise en œuvre via un Framework de modélisation et de simulation basées sur des agents (ABMS). La création de divers ABMS intégrant différentes stratégies visant à endiguer la propagation d'une épidémie. Enfin, le développement d'une application web permettant l'analyse comparative des divers ABMS créés et de leurs stratégies respectives.

L'objectif principal de ce stage réside dans l'analyse de l'efficacité des mesures de protection mises en place au sein d'une population pour contrer une épidémie, afin de démontrer qu'il n'est pas nécessaire d'imposer systématiquement certaines mesures à chaque individu pour contenir la propagation d'une maladie infectieuse. En effet, certaines de ces mesures peuvent restreindre la liberté individuelle.

Au cours d'une période de 10 semaines de stage, comment ai-je réussi à structurer mon travail de manière efficiente et à répondre aux attentes de M. Costin Bădică ?

Dans un premier temps, nous aborderons la présentation générale du sujet, dont le contexte, l'expression des besoins du projet et l'environnement de travail. Ensuite, nous examinerons le modèle spatial multi-agent SIR développé par Monsieur Bădică et ma démarche pour son implémentation. Par la suite, nous analyserons les diverses stratégies déployées pour contrer la propagation d'une épidémie. Enfin, je procéderai à la présentation de l'application web réalisée et de son utilité.

# 1 Contexte du projet

## 1.1 L'université de Craiova

### 1.1.1 Historique

Afin de valider mon Bachelor Universitaire de Technologie Informatique, j'ai effectué mon stage au sein de l'Université de Craiova. Elle fut fondée en 1947 et comptait au départ 4 instituts. C'est la plus grande université de la province d'Olténie (le Sud-ouest de la Roumanie) et du Comté de Dolj. Elle fait partie de l'Association des Universités Européennes et intègre le système de crédits ECTS. En 2024, elle est classée 18e université de Roumanie.

L'université compte s'est développée au fil du temps pour compter aujourd'hui 17 facultés, accueillant plus de 25 000 étudiants et proposant une variété de programmes d'études dans divers domaines.

L'Université de Craiova bénéficie également d'une histoire riche en matière de recherche et d'innovation. Les chercheurs de l'université ont contribué significativement, au fil des années, dans des domaines tels que l'agriculture, la médecine vétérinaire, l'ingénierie, la chimie et la physique.

Par ailleurs, l'Université de Craiova a joué un rôle primordial dans le domaine culturel et social de la ville de Craiova. En accueillant divers événements culturels et scientifiques, elle a formé un grand nombre de professionnels qualifiés ayant apporté une contribution majeure au développement régional. Ainsi, elle est désormais considérée comme l'une des institutions d'enseignement supérieur les plus prestigieuses en Roumanie.

### Symbolique

La devise de l'université est : « *Vita sine litteris mors est* », la vie sans apprentissage est la mort. Son logo (voir Figure 1), affiché sur son site web ainsi que sur les documents officiels, présente un blason avec 3 parties distinctes :

- En bas à gauche, un lion avec une étoile, symbole de la province historique d'Olténie ;
- En bas à droite, un livre ouvert, symbole du savoir et de l'enseignement ;
- En haut, les armoiries de Roumanie, l'aigle d'or avec un sceptre et une épée, symboles de souveraineté.



Figure 1 – Logo de l'Université de Craiova

### 1.1.2 Groupe de recherche

Au sein de l'Université de Craiova, un grand nombre de professeurs y exercent, mais c'est avec Monsieur Costin Bădică, enseignant en informatique et expert en intelligence artificielle,

systèmes multi-agents, génie logiciel et systèmes distribués, ayant à son actif de nombreux projets, que j'ai eu l'opportunité de travailler sur mon projet.

Il convient de mentionner que nous étions trois étudiants de l'IUT du Limousin : Noah STAPLE, Nathan PINHEIRO et moi-même. Bien que nous ayons eu le même encadrant pédagogique, nos projets étaient distincts. Nos réunions se déroulaient conjointement, formant ainsi un groupe de recherche composé de quatre membres.

### 1.1.3 Recherches sur la modélisation des maladies infectieuses

La recherche sur la modélisation mathématique de la propagation des maladies infectieuses et contagieuses a commencé il y a presque un siècle. L'objectif était le développement de modèles dynamiques capables de capturer et d'expliquer les différents facteurs pouvant influencer l'intensité et la durée de la maladie au sein d'une population donnée d'individus.

Un cas bien connu qui a menacé l'humanité au début du XXe siècle est la « grippe pandémique de 1918 » également connue sous le nom de « grippe espagnole ». Elle a duré plus de 2 ans, causant la mort de plus de 500 millions de personnes. Un autre cas très actuel, vécu actuellement par l'humanité tout entière, est la « maladie du coronavirus 2019 » ou « COVID-19 », maladie respiratoire et vasculaire contagieuse, également connue sous le nom de « syndrome respiratoire aigu sévère coronavirus 2 » ou « SRAS-CoV-2 ». Elle a suscité un énorme intérêt de recherche dans les sciences biomédicales et informatiques, ainsi que la nécessité de développer des modèles de simulation détaillés de la dynamique de propagation de la maladie, y compris la conception et l'analyse de diverses stratégies et politiques d'atténuation.

Ainsi, Costin Bădică, Amelia Bădică, Maria Ganzha, Mirjana Ivanovic et Marcin Paprzycki ont rédigé un article [1] décrivant leur élaboration d'un système multi-agent pour la modélisation et la simulation des stratégies de gestion de la propagation des épidémies. Le cœur de l'approche proposée est un système discret stochastique générique de Susceptible-Infecté-Récupéré spatial, en utilisant la plateforme d'agent GAMA. Son objectif est d'évaluer l'effet des mesures prophylactiques et de limitation de la mobilité sur l'impact et l'ampleur de la propagation des épidémies. Du fait de son abstraction soignée et sa définition précise, ce modèle se révèle particulièrement adapté comme point de départ pour une extension future ainsi qu'une application à des modèles plus détaillés relatifs à des problèmes spécifiques.

## 1.2 Expression des besoins

Mon travail de recherche a consisté à analyser et à reproduire le système multi-agent élaboré en utilisant une autre plateforme et à proposer de nouvelles stratégies pour gérer la propagation de l'épidémie. Je me suis également fixé l'objectif de produire une application web d'analyse de ces différentes stratégies.

### 1.2.1 Notions importantes

Les modèles compartimentaux trouvent leurs origines dans les premiers travaux de Kermack et McKendrick. Ils sont utilisés par les épidémiologistes et les mathématiciens pour abstraire le processus de propagation des maladies infectieuses et simplifier leur analyse mathématique. Ces modèles proposent de structurer le processus de propagation des épidémies au sein de la population par des compartiments, en fonction de l'état des individus de la population. Des exemples de compartiments sont : Susceptible, Infecté et Rétabli, généralement étiquetés comme S, I et R. Ces

modèles sont définis par des équations différentielles ou des équations aux différences qui capturent la dynamique du transfert de population entre les compartiments, selon le degré ou le stade de l'infection. Un tel modèle peut être utilisé pour étudier théoriquement ou expérimentalement la propagation de l'épidémie dans le temps et dans l'espace géographique.

La modélisation et la simulation basées sur les agents (ABMS : Agent Based Model Simulation) constituent une approche solidement établie offrant diverses possibilités d'application dans les systèmes naturels, incluant notamment la simulation des processus de diffusion stochastiques dans des environnements continus et non-linéaires. L'ABMS permet aux experts du domaine de choisir entre différents niveaux de granularité de modélisation, qu'il s'agisse du macroscopique, du méso-scopique ou du microscopique. Cette souplesse autorise une focalisation sur divers éléments du système cible en trouvant un juste-milieu entre capturer les détails et exploiter l'efficacité de la modélisation. Il est pertinent de souligner que les techniques et outils ABMS ont joué un rôle crucial dans le développement des simulations de propagation virale via des plateformes ABMS innovantes telles que GAMA, MESA et JS-son.

Un agent est une entité autonome qui peut interagir avec son environnement et avec d'autres agents. Les agents sont généralement définis par leurs attributs (par exemple, la position, la vitesse, l'énergie), leurs comportements (par exemple, se déplacer, manger, communiquer) et leurs interactions avec les autres agents et leur environnement.

### 1.2.2 Compréhension du modèle multi-agent SIR et création d'un ABMS

Ma première tâche a été d'analyser et comprendre le modèle multi-agent spatial générique Susceptible-Infecté-Rétablissement (SIR) pour étudier la propagation des épidémies au sein d'une population, réalisée par Costin Bădică et son équipe de recherche [1]. Mon objectif était de mettre en œuvre un ABMS reprenant ce modèle, tout en le personnalisant à ma propre démarche.

En effet, le modèle proposé segmente la population d'agents en trois catégories (SIR) : les agents susceptibles, les agents infectés, et aucun individu retiré au départ. Au cours de la simulation, la population évolue selon des probabilités déterminées : les individus susceptibles ont une probabilité de contracter la maladie, les individus infectés ont une probabilité de transmission, et les individus infectés ont une probabilité de passer dans la catégorie retirée (soit par décès, soit par immunisation). Un dispositif de contrôle est instauré pour surveiller le taux moyen d'individus infectés à chaque étape de la simulation. Si ce taux dépasse un seuil prédéfini, tous les agents voient leur vitesse réduite. La simulation prend fin lorsque tous les agents infectés ont été guéris.

### 1.2.3 Ajout de nouvelles stratégies contre une épidémie

Ma deuxième tâche a consisté à ajouter des nouvelles stratégies sur la simulation multi-agent (ABMS) du modèle spatial SIR que j'ai mis en œuvre, en m'appuyant sur les travaux de recherche de M. Bădică [1].

Tel qu'énoncé antérieurement, le dispositif de contrôle restreint la mobilité de l'ensemble de la population. Cependant, il convient d'examiner la nécessité de cette restriction généralisée. C'est pourquoi j'ai élaboré différentes stratégies sur mon ABMS reprenant le modèle spatial SIR de M. Bădică, mettant en place ainsi de nouveaux ABMS dont un avec un système de zones statiques et un autre avec un système de création de zones dynamiques pour contenir la propagation de l'épidémie. L'objectif est d'évaluer la possibilité de limiter moins d'individus au sein d'une population tout en obtenant le même effet sur la réduction de la propagation du fléau.

De plus, des ABMS ont été mis au point intégrant le port du masque afin d'étudier son incidence sur la diffusion d'une épidémie.

### 1.2.4 Application web

Ma troisième tâche a été de concevoir une application Web d'analyse regroupant l'ensemble des ABMS avec leurs stratégies respectives, afin de comparer les résultats obtenus lors des différentes simulations à travers des graphiques et des statistiques.

Cette application permet de personnaliser les paramètres des modèles des ABMS afin de pouvoir effectuer plusieurs séries d'essais pour observer comment se propage une épidémie en fonction de divers paramètres et mesures.

## 1.3 Organisation et outil

### 1.3.1 Environnement de travail

Pour ce qui est du matériel, je n'ai eu que mon ordinateur portable à ma disposition, car mon maître de stage m'a conseillé de l'amener et de l'utiliser à Craiova, pour des raisons pratiques.

Dans le cadre de ce projet de recherche, M. Bădică m'a envoyé son article de recherche sur le modèle multi-agent spatial SIR et la modélisation d'une propagation d'épidémie [1]. Cependant aucune contrainte ne m'a été imposée en termes d'outils pour l'environnement de travail ou les outils techniques. Ainsi, vais détailler mes choix relatifs aux outils utilisés pour la réalisation de mon projet.

En premier lieu, concernant l'éditeur de code / IDE (*Integrated Development Environment*), j'ai opté pour l'utilisation de Visual Studio Code (VS Code) car il est *Open Source* et c'est celui que j'ai le plus utilisé durant ma deuxième de BUT informatique. Ce choix s'est justifié par ses nombreux avantages, dont deux principaux qui se révèlent essentiels : le support intégré pour Git et la possibilité d'intégrer des extensions permettant d'ajouter de nouveaux langages, thèmes et surtout des fonctionnalités de débogage qui peuvent se révéler extrêmement utiles faire gagner un temps considérable.

```
name = "Mohamed"
if name == "Mohamed"    Expected ":"  
|   print("Hello Mohamed")
```

Figure 2 – Exemple de fonctionnalité de débogage de VS Code

Concernant la gestion des versions de mon développement, j'utilise une plateforme web d'hébergement et de gestion de logiciels, reposant sur le système de contrôle de version décentralisé Git : GitHub. C'est la plateforme avec laquelle je suis le plus familier, elle se distingue par sa simplicité d'utilisation et offre une représentation visuelle claire des branches et des commit. De plus, il m'arrive d'utiliser l'application GitBash ou l'IDE VS Code pour versionner mon projet sur cette plateforme.

A propos de l'organisation de mon travail, j'utilise l'outil de gestion de projet n°1 : Trello, qui favorise une collaboration efficace en équipe. Même si je mène ce projet en solitaire, cet outil me permet de structurer aisément mes tâches en fixant des objectifs précis, ce qui facilite le suivi de l'avancement du projet. De plus l'ajout de délais m'aide à évaluer si je respecte les échéances et à réajuster mon planning en conséquence. Ainsi, Trello encourage la mise en œuvre d'une approche agile du travail.

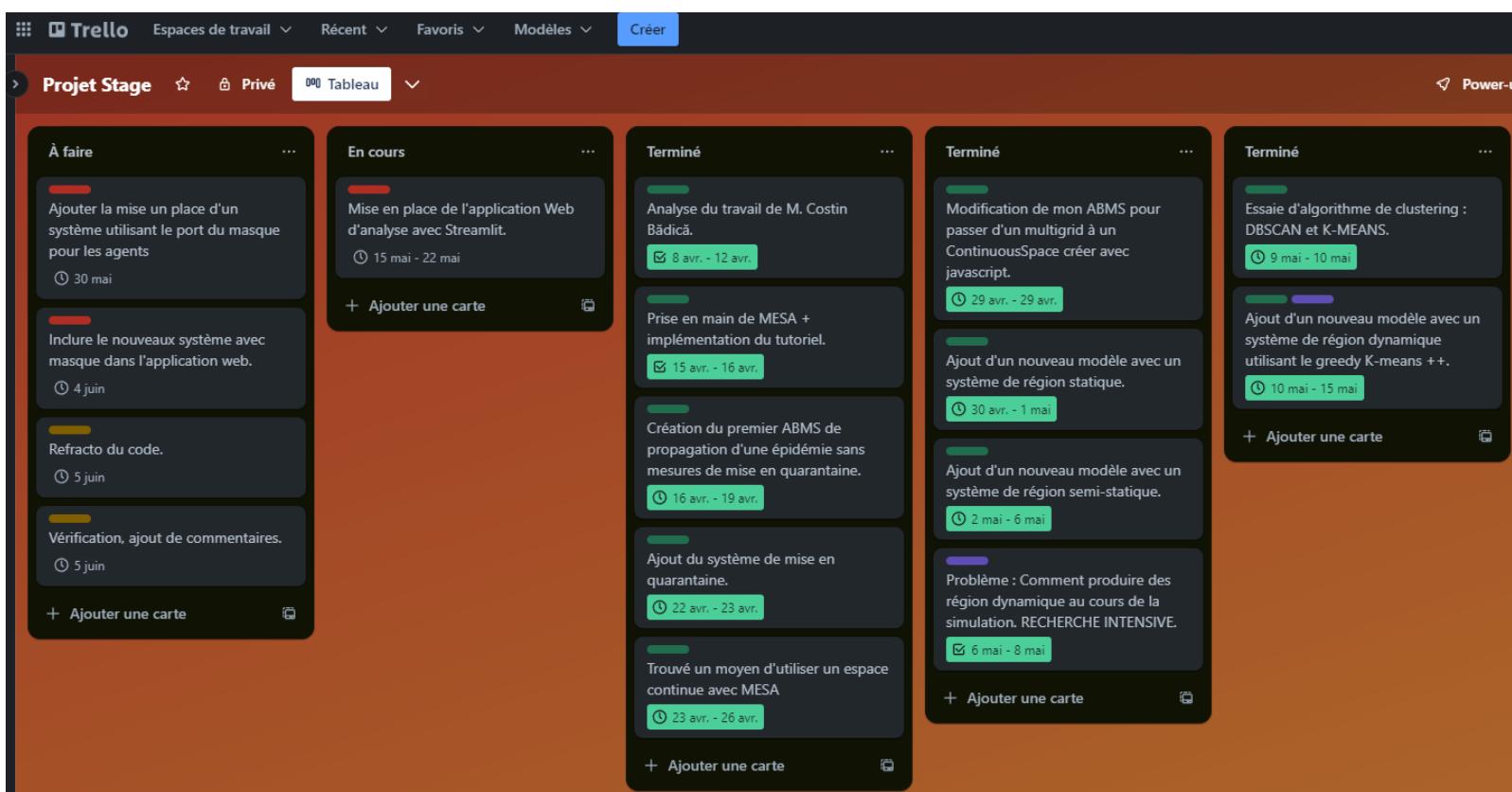


Figure 3 – Outils de gestion de projet : Trello

En ce qui concerne le suivi de mon travail, nous avions l'habitude d'organiser chaque semaine une réunion en personne à l'Université de Craiova avec M. Bădică afin de discuter de mes progrès et des ajustements éventuels à apporter. De plus, une réunion en ligne était planifiée avec Mme Onete pour échanger également sur ma progression.

### 1.3.2 Outils techniques

Il a fallu que je me décide rapidement sur le choix d'un Framework pour réaliser mes ABMS. M. Bădică m'a recommandé d'utiliser MESA ou JS-son.

#### 1.3.2.1 Les frameworks MESA et JS-son pour la modélisation et la simulation basées sur les agents

MESA et JS-son sont deux Frameworks de modélisation et de simulation basés sur les agents. Les deux sont open-source, ce qui signifie que tout le monde peut accéder à leur code source, le modifier et le distribuer. Les agents dans les modèles peuvent représenter des individus, des organisations, des véhicules, des animaux, etc.

MESA est un Framework en Python sous licence Apache2, tandis que JS-son est un Framework en JavaScript. Les deux sont open-source.

MESA fournit une large gamme d'outils pour la modélisation et la simulation, ce qui en fait un choix populaire pour les chercheurs. Il permet aux utilisateurs de créer des modèles en utilisant une syntaxe Python simple et intuitive, et fournit des méthodes pour collecter et analyser les données de simulation. MESA est également extensible, ce qui signifie que les utilisateurs peuvent ajouter leurs propres bibliothèques et modules pour répondre à des besoins spécifiques.

JS-son se distingue par sa capacité à intégrer des données sonores dans les modèles, ce qui peut être utile pour simuler des phénomènes tels que la propagation du son dans un environnement. Il permet aux utilisateurs de créer des modèles en utilisant un langage de programmation visuel basé sur des blocs, ou en écrivant du code JavaScript. Les modèles peuvent être exécutés dans le navigateur web, ce qui facilite le partage et la collaboration.

### 1.3.2.2 Choix des outils techniques

En tant que choix populaire chez les chercheurs et étant moi-même engagé dans la recherche, j'ai donc opté pour l'utilisation de MESA afin de modéliser la propagation d'une épidémie.

De cette façon, je vais me servir de Python, un langage de programmation interprété, multiparadigme et open-source. Reconnu pour sa syntaxe simple et facile à lire, ainsi que pour sa grande flexibilité et ses nombreuses bibliothèques et Framework, Python est largement utilisé dans divers domaines tels que la science des données, l'intelligence artificielle, le développement web et la conception de logiciels.

En outre, j'ai dû utiliser Javascript, un langage de programmation orienté objet principalement orienté client, qui me permet de manipuler le DOM (Document Object Model) qui comprend les éléments, les propriétés et l'apparence visuelle.

En définitive, j'ai choisi d'utiliser Streamlit, un Framework Python open source, car il offre la possibilité de concevoir aisément des applications web interactives d'analyse. Contrairement à d'autres Frameworks populaires tels que Flask, qui offrent un environnement flexible nécessitant des développeurs qu'ils définissent des itinéraires, des modèles et autres éléments pour avoir un contrôle accru sur le comportement de l'application, Streamlit se distingue par sa conception extrêmement conviviale. Il requiert un minimum de configuration et de code répétitif, ce qui en fait l'outil parfait pour mon application web développée au cours des dernières semaines de mon stage.

## 2 Mise en place du premier ABMS

### 2.1 Le travail de recherche existant

La modélisation de la propagation d'une épidémie au sein d'une population requiert en premier lieu une phase de recherche mathématique. Comme mentionné précédemment, j'ai basé mes travaux sur les recherches fournies par M. Bădică sur ce sujet, en adoptant exactement les mêmes formules et valeurs que lui, comme il me l'avait fortement recommandé. Ainsi, à partir de l'article de M. Bădică [1], je vais résumer le fonctionnement du modèle spatial SIR, la modélisation de la propagation d'infection, la gestion de l'infection et la représentation de ce modèle en système multi-agent.

#### 2.1.1 Le modèle spatial SIR de base

Le modèle SIR (Susceptible-Infected-Removed) est un modèle mathématique utilisé pour décrire la dynamique de propagation d'une maladie infectieuse dans une population. Il divise la population en trois compartiments :

Les individus susceptibles (S) : ce sont les personnes qui peuvent être infectées par la maladie, et la population d'agent susceptible à l'instant  $t$  est représenté par  $S_t$ . Les individus infectés (I) : ce sont les personnes qui ont été infectées par la maladie et qui peuvent la transmettre à d'autres

personnes qui sont susceptibles, et la population d'agent infecté à l'instant  $t$  est représenté par  $I_t$ . Les individus « retirés » (R) : ce sont les personnes qui ont été infectées par la maladie et qui ont guéri, devenu résistant ou sont mortes, et la population d'agent « retiré » est représenté par  $R_t$ .

L'objectif du modèle spatial SIR est de capturer les dynamiques de population dans l'espace et le temps, avec la variation du nombre de personnes susceptibles, infectées, « retirées » dans la population entre l'instant  $t$  et l'instant  $t + \Delta t$ , avec :

$$\Delta S = S_{t+\Delta t} - S_t$$

$$\Delta I = I_{t+\Delta t} - I_t$$

$$\Delta R = R_{t+\Delta t} - R_t$$

Ainsi la population totale au temps  $t$  est  $N_t = S_t + I_t + R_t$ .

Le modèle décrit comment la maladie se propage et évolue dans le temps et l'espace. La propagation est déterminée par une fonction d'infection  $F$ , qui dépend du nombre de d'individus susceptibles et infectés. Une partie des infectés devient « retirés », (guérie ou décédée), à chaque étape. Les variations des populations sont suivies de manière discrète à chaque intervalle de temps constant.

### 2.1.2 Modélisation de la propagation de l'infection

Dans ce modèle spatial, chaque individu a une position physique, permettant de capturer la propagation locale de la maladie. Les individus peuvent changer de lieu à chaque étape, ce qui modélise à la fois la dimension spatiale et temporelle de la propagation.

- **Localisation** : La position d'un individu est notée  $i.x$  et une fonction de distance  $d$  mesure la proximité : distance Euclidienne.
- **Voisinage** : Le voisinage  $V(x)$  d'une position  $x$  est défini comme les lieux à l'intérieur d'un rayon  $\epsilon$ , noté  $B(x, \epsilon)$ .
- **Infectés à proximité** : Pour un individu  $j$ ,  $I(j)$  est l'ensemble des infectés dans son voisinage  $B(j.x, \epsilon)$ .
- **Probabilités** :  $p_i$  est la probabilité qu'un individu  $i$  transmette l'infection, et  $q_i$  est la probabilité qu'il soit infecté s'il est proche d'un infecté.
- **Atténuation spatiale** : La fonction  $\phi(x)$  modélise l'atténuation de la transmission avec la distance  $x$ .

Ainsi, la probabilité qu'un individu susceptible  $j$  soit infecté est donnée par la formule ci-contre :

$$q_j \cdot \left( 1 - \prod_{i \in I(j)} (1 - p_i \cdot \phi(d_{ij})) \right)$$

Cette probabilité est bien définie dans l'intervalle  $[0, q_j]$  et elle est nulle si aucun infecté n'est à proximité.

### 2.1.3 Gestion de l'infection

L'approche proposée pour la gestion de la propagation de l'infection suppose trois étapes : l'évaluation de la situation, la détection et l'atténuation.

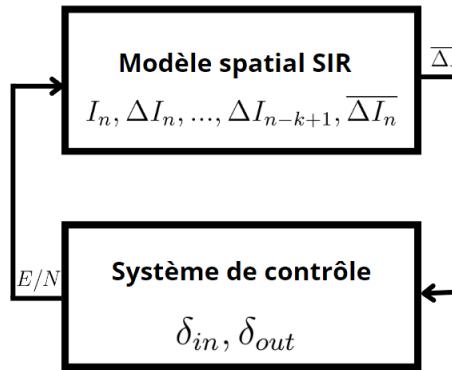


Figure 5 – Diagramme du système de contrôle

#### Évaluation de la situation

- Utilise des capteurs sociaux pour obtenir des informations, notamment via le traçage des contacts.
- Suit la variation du nombre d'infections et estime le taux de propagation.
- Utilise la moyenne mobile des dernières étapes pour estimer le taux d'infection avec :  $\Delta I_n = I_n - I_{n-1}$  pour  $n \geq 1$  ,  $\overline{\Delta I_n} = \frac{\sum_{i=1}^k \Delta I_{n+i-1}}{k}$  pour  $k \geq 1$  ou  $I_n$  représente le nombre de personne infecté à l'étape  $n$ . Et  $k$  définie comme étant proportionnelle à  $T_r$ , le temps moyen pendant lequel une personne reste infectée avant de guérir, en utilisant une constante  $k^{mem} > 0$ , comme suit :  $k = k^{mem} \cdot T_r$

#### Détection

- Compare le taux d'infection par rapport à un seuil fixé  $\delta_{in}$ .
- Si le taux dépasse ce seuil, le système passe en phase d'urgence (atténuation).

#### Atténuation

- Déclare une « situation d'urgence » avec une limitation de la mobilité.
- Continue de surveiller le taux d'infection et compare à un second seuil  $\delta_{out}$  . Si le taux passe en dessous de ce seuil (qui est négatif pour garantir une diminution), le système revient à la normale.
- La durée minimale de l'état d'urgence est proportionnelle au temps de retrait  $T_r$  .

Les seuils  $\delta_{in}$  et  $\delta_{out}$  sont ajustés après chaque sortie de l'état d'urgence pour s'adapter à la situation actuelle. Les valeurs initiales des seuils sont définies en fonction de la taille de la population.

## 2.1.4 Représentation multi-agent

La mise en correspondance du modèle avec une représentation multi-agent est simple. Fondamentalement, chaque individu est représenté comme un agent. Le compartiment auquel chaque individu est assigné est capturé comme l'état de l'agent, c'est-à-dire  $S$ ,  $I$  ou  $R$ . Initialement, la plupart des agents sont dans l'état  $S$ , quelques-uns sont dans l'état  $I$ , et aucun n'est dans l'état  $R$ . À la fin de la simulation, lorsque le processus se stabilise, chaque agent est soit dans l'état  $S$  soit dans l'état  $R$ . Donc, l'état  $I$  est un « état transitoire » des agents. De plus, si après le processus de stabilisation, tous les agents sont dans l'état  $R$ , cela signifie que l'épidémie a atteint toute la population.

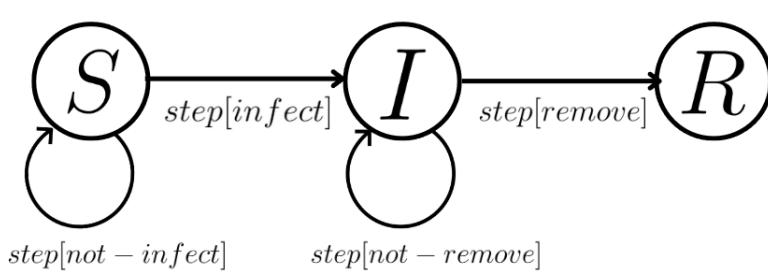


Figure 6 – Diagramme de transition d'état d'un individu.

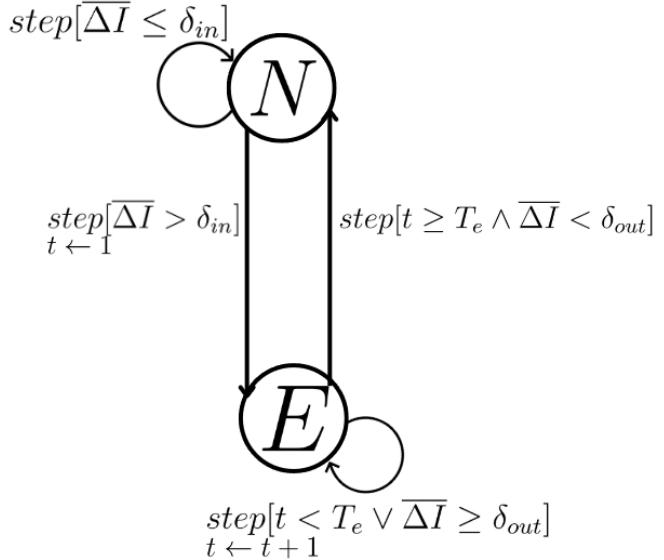


Figure 7 – Diagramme de transition d'état du contrôleur.

Les transitions d'état des agents sont modélisées comme un automate fini (Fig.6), avec des transitions synchronisées à chaque étape de la simulation. Ces transitions sont stochastiques et les probabilités sont déterminées par les paramètres du modèle.

- **Propagation de l'infection :** Un agent infecté envoie un « message d'infection » à tous les agents susceptibles dans son voisinage avec une probabilité calculée selon la probabilité qu'un individu susceptible  $j$  soit infecté.
- **Passage de I à R :** Un agent infecté passe à l'état  $R$  avec une probabilité  $\gamma$ .

Les agents sont mobiles, reflétant la mobilité des individus dans la population, et cette mobilité est intégrée dans le comportement spatial des agents.

Le contrôleur du système est également modélisé comme un automate fini (Fig.7), avec deux états principaux : **État N (normal)** : Situation normale du système. **État E (urgence)** : Situation d'urgence déclenchée lorsque  $\Delta I > \delta_{in}$ . Le retour à l'état normal se fait lorsque  $\Delta I < \delta_{out}$  et que le temps  $t$  écoulé depuis le début de l'urgence est supérieur ou égal à  $T_e$ , la durée minimale de l'état d'urgence (proportionnelle à la durée moyenne de la maladie  $T_r$ ).

La variable  $t$  représente le temps écoulé, (en nombre d'étapes), depuis le déclenchement de l'état d'urgence. Elle est initialisée à 1 lors de la transition de N à E et incrémentée à chaque étape pendant l'état d'urgence.

## 2.2 Développement

### 2.2.1 Structure du Framework MESA

MESA propose une architecture avec principalement trois niveaux pour les systèmes multi-agents : le niveau de l'agent, le niveau de l'environnement dans lequel les agents interagissent : le modèle et le niveau de l'interface utilisateur permettant d'interagir avec le modèle : le serveur, (voir annexe 1 pour l'architecture complète).

Cette architecture offre une flexibilité et une extensibilité pour la création de modèles à base d'agents complexes et personnalisés en utilisant Python. Ainsi, chaque ABMS réalisé sera décomposé en 3 fichiers python : *agent.py*, *model.py* et *server.py*.

#### 2.1.2.1 *agent.py*

Au sein de ce niveau, j'ai mis en œuvre la classe *Host* qui représente les agents, soit les entités autonomes constituant le modèle. Les agents sont dotés d'un *id*, des booléens *is\_susceptible*, *is\_infected* et *is\_removed* pour déterminer leur compartiment dans le modèle SIR, ce qui leur permet d'effectuer certaines actions et de se voir attribuer une couleur spécifique lors de la simulation : vert pour susceptible, rouge pour infecté et bleu pour « retiré ». De plus, ils possèdent des attributs tels que *speed* pour leur vitesse, *neighbors* pour la position de leurs voisins dans un rayon donné et *pos* pour leurs coordonnées au sein de l'environnement (modèle).

C'est au sein de cette classe que le comportement des agents dans la simulation est défini. Ils se déplaceront de manière aléatoire en fonction d'un angle compris entre 0 et 360 degrés, d'une direction et d'une vitesse afin de simuler le déplacement d'un individu réel.

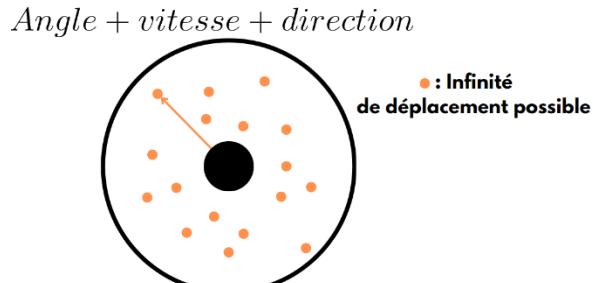


Figure 8 – Déplacement d'un agent.

Les agents infectés pourront transmettre la maladie aux agents susceptibles pour qu'ils deviennent à leurs tours infectés, selon les probabilités  $p$  et  $q$ . Les agents infectés pourront également devenir des agents « retirés » selon le taux de guérison  $\gamma$ .

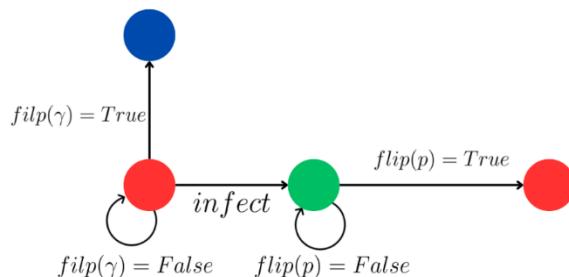


Figure 9 – Comportement des agents. (Rouge : infecté / vert : susceptible / bleu « retiré ».  $\gamma$  : taux de guérison /  $p$  : probabilité qu'un agent infecté a transmis sa maladie à un agent susceptible et que l'agent susceptible contracte la maladie.  $flip()$  : détermine si un événement se réalisera.

Concernant l'algorithme de la méthode infect ci-dessous :

```
def infect(self) -> None:
    range = truncated_gauss(self.model.infect_range, self.model.infect_range / 2.0)
    self.neighbors = self.model.space.get_neighbors(self.pos, range, True)
    if self.neighbors:
        for susceptible_agent in self.neighbors:
            p = 1.0
            if susceptible_agent.is_susceptible:
                neighbors_of_susceptible_agent = self.model.space.get_neighbors(susceptible_agent.pos, range)
                if neighbors_of_susceptible_agent:
                    for agent in neighbors_of_susceptible_agent:
                        if agent.is_infected:
                            d = susceptible_agent.distance_to(agent)
                            p = p * (1.0 - self.model.p_inf * self.phi(d))
            p = self.model.q_inf * (1.0 - p)

            if flip(p):
                susceptible_agent.is_susceptible = False
                susceptible_agent.is_infected = True
                susceptible_agent.color = "red"
```

Figure 10 – Algorithme de la méthode infect.

Elle examine, pour chaque agent infecté, les agents présents dans leur zone d'infection, et les recueille dans une liste. Ensuite, elle parcourt la liste et crée une autre liste des agents susceptibles. Ainsi, tous les agents susceptibles se trouvant dans la zone d'infection d'un agent infecté sont identifiés. Ensuite, pour chacun de ces agents susceptibles, la méthode récupère les agents présents dans leur zone d'infection et évalue la probabilité qu'un individu susceptible soit infecté.

```
for agent in neighbors_of_susceptible_agent:
    if agent.is_infected:
        d = susceptible_agent.distance_to(agent)
        p = p * (1.0 - self.model.p_inf * self.phi(d))
p = self.model.q_inf * (1.0 - p)
```

$$q_j \cdot \left(1 - \prod_{i \in I(j)} (1 - p_i \cdot \phi(d_{ij}))\right)$$

Figure 11 – Implémentation de la formule d'infection.

Par conséquent, plus il y a d'agents infectés dans la zone d'infection d'un agent susceptible, plus ce dernier a de chances de devenir un agent infecté. Une fois cette probabilité calculé,  $flip()$  va déterminer si cette probabilité aura lieu ou non.

## Illustration

L'agent infecté rouge au centre, figure 12, rassemble tous les agents présents dans sa zone d'infection. Ensuite, il conserve les agents susceptibles (agents verts).

Toutefois, ces agents susceptibles doivent également rassembler tous les agents de leur zone d'infection respective. Pourquoi donc cette procédure ? Ne serait-il pas plus judicieux que chaque agent infecté rassemble l'ensemble des agents de leurs zones respectives, puis calcule la probabilité qu'un agent susceptible soit infecté et applique cette probabilité à chaque agent susceptible de la zone ?

Il est à noter que la proximité de deux agents dans la zone d'infection d'un tiers ne garantit pas leur présence simultanée dans la zone d'infection des autres agents.

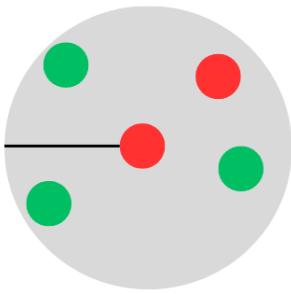


Figure 12 – Illustration méthode infect 1.

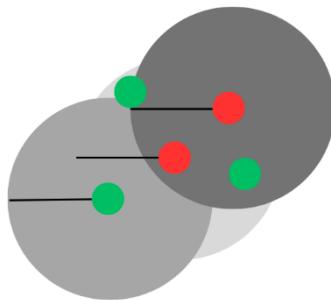


Figure 13 – Illustration méthode infect 2

En effet sur la figure 13, l'agent susceptible situé en bas à gauche sera exposé à un seul agent infecté dans sa zone d'infection, réduisant ainsi sa probabilité d'être infecté. En revanche, l'agent susceptible en bas à droite devra composer avec la présence de deux agents infectés dans sa zone d'infection.

### Remarque

J'aurais pu simplifier davantage cette méthode en l'appliquant à chaque agent susceptible au lieu des agents infectés, comme l'a fait M. Bădică. Cela m'aurait permis de cibler directement les agents se trouvant dans la zone d'infection des agents susceptibles et de calculer la probabilité qu'ils contractent la maladie s'il y a des agents infectés à proximité. Cependant, avec un grand nombre d'agents dans le modèle basé sur l'agent (ABMS), la simulation était trop gourmande en termes de performances, notamment avec le système de régions dynamiques mis en place plus tard. En effet, le nombre d'agents susceptibles est bien plus élevé que celui des agents infectés. De plus, le nombre d'agents infectés reste toujours relativement modéré par rapport aux susceptibles car ces derniers peuvent devenir des agents « retirés » (soit décédés de la maladie soit immunisés).

#### 2.1.2.2 model.py

Le modèle représente le cadre dans lequel les agents interagissent. J'ai élaboré la classe *SIRModel* qui requiert un ensemble de paramètres pour son instantiation dont *width / height* pour les dimension du modèle; *Number\_S* : le nombre d'agent susceptible ; *Number\_I* : le nombre d'agent infecté ; *p\_inf\_high / q\_inf\_high* : la probabilité de transmission / contraction de la maladie sans mesures de protections ; *p\_inf\_high\_after / q\_inf\_high\_after* la probabilité de transmission / contraction de la maladie après la fin d'un état d'urgence ; *p\_inf\_low / q\_inf\_low* : la probabilité de transmission / contraction de la maladie avec mesures de protections; *gamma* : taux de guérison ; *wandering\_speed\_high / wandering\_speed\_low* : vitesse de déplacement des agents sans et avec système de contrôle; *infect\_range* : rayon pour définir la zone d'infection pour chaque agent; *max\_itérations* : le nombre maximum d'itérations pour la simulation ; *alpha* : constante pour la fonction d'atténuation spatial ; *coef\_quarantine/amplif\_quarantine/atten\_quarantine* : coefficients utilisés pour la durée de l'état d'urgence.*coef\_threshold\_in/coef\_threshold\_out* : coefficient utilisé pour calculer les seuils d'entrée et sorties d'un état d'urgence. (Ces derniers seront fournis via le fichier *serveur.py*).

Avec ces paramètres, cette classe initialisera l'environnement, les agents et leur position au sein de ce dernier. Elle initialisera également l'ensemble des paramètres qui seront utilisé pour les méthodes des agents et les méthodes des systèmes de contrôles. Ainsi, la classe *SIRModel* occupera une position centrale dans la simulation, étant responsable de diverses tâches telles que le déplacement des agents, la collecte des données, la mise à jour des données et l'exercice d'un contrôle sur celles-ci. Elle intègrera notamment des mécanismes visant à réguler la propagation de l'épidémie et à mettre fin à la simulation. Le premier mécanisme est la réduction considérable de la vitesse des agents tout au long d'un état d'urgence. En effet, comme les mouvements des agents dépendent de leur vitesse, figure 8, si agent voit sa vitesse considérablement réduite il ne pourra se déplacer que très peu simulant ainsi une mise en confinement ou les déplacements sont très moindre.

Jusqu'à ce que max\_iteration atteint ou que nb agents infectés = 0.  
Le modèle va permettre à chaque étape de la simulation :

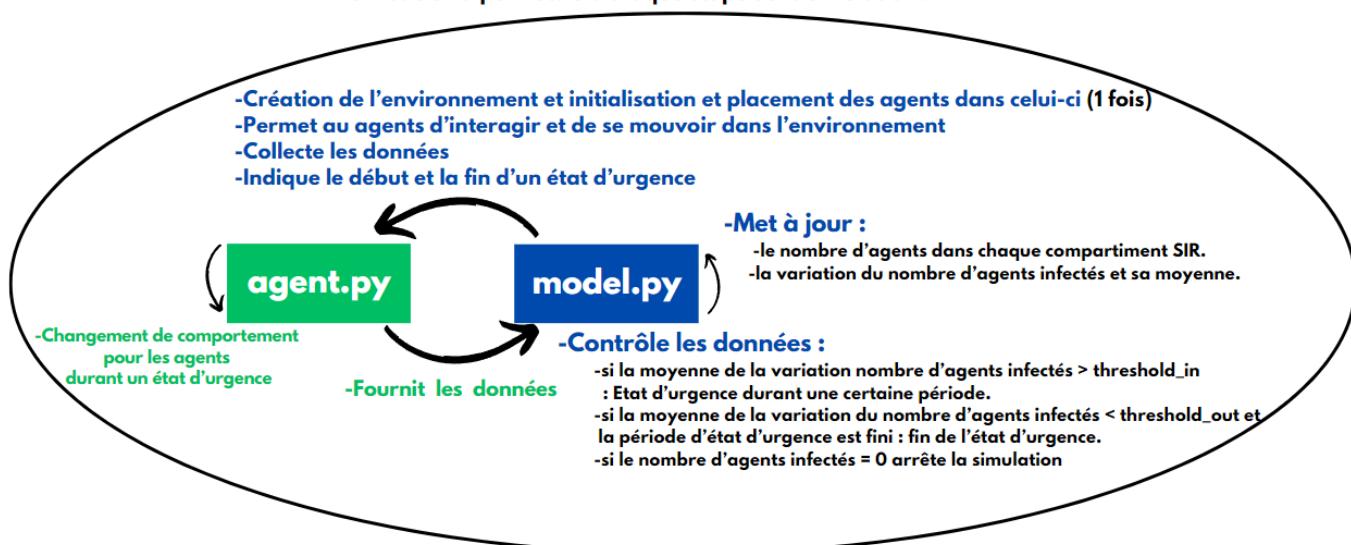


Figure 14 – Rôle du modèle.

Pour la création de l'environnement MESA en propose deux types : les espaces avec grille et l'espace continu. Les espaces avec grille (Fig.15) permettent de placer un ou plusieurs agents sur chaque cellule. Dans cet espace, les agents se déplacent de cellule en cellule, captant ainsi les voisines (Fig. 16).

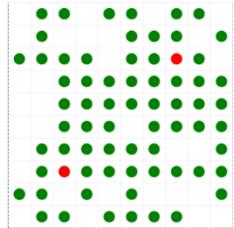


Figure 15 – Espace avec grille.

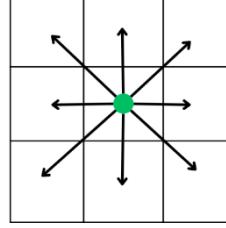


Figure 16 – Déplacement dans un espace avec grille.

Dans l'espace continu (Fig.15), les agents peuvent se mouvoir en fonction d'un angle, direction et vitesse, offrant ainsi davantage de possibilités de déplacement. Par conséquent, j'ai opté pour un espace continu afin d'obtenir une représentation plus fidèle des mouvements des agents et une simulation plus réaliste. Il convient de noter que MESA offre la possibilité de rendre les environnements toroïdaux, c'est-à-dire que les agents peuvent passer d'un bord à l'autre (haut en bas

et gauche à droite) dans l'espace continu. Toutefois, pour maintenir la cohérence d'une simulation réaliste, j'ai choisi de ne pas activer cette option.

Toutefois, ces environnements requièrent l'utilisation d'un module de visualisation pour être visualiser sur le serveur. MESA a seulement d'implémenté un unique module de visualisation, le CanvaGrid, qui est limité aux espaces dotés de grilles. Par conséquent, j'ai développé en JavaScript, (première et dernière utilisation), et en Python mon propre module de visualisation, en m'inspirant du CanvaGrid et de sa documentation, afin de pouvoir visualiser mon espace continu.

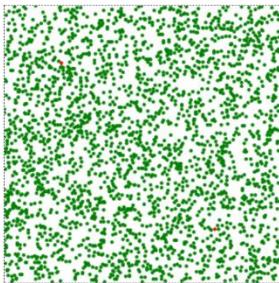


Figure 17 – Espace continu.

### 2.1.2.3 Serveur.py

Le serveur procédera à l'instanciation de la classe SIRModel en utilisant les paramètres (Fig. 16) dont les valeurs ont été extraites des travaux de Monsieur Bădică [1]. Ces paramètres seront ensuite employés dans diverses méthodes pour le calcul des probabilités et la gestion de la propagation de l'épidémie. **Remarque :** dans son article, Monsieur Bădică a utilisé un environnement de 120x120 avec la plateforme ABMS GAMA, cependant, avec le Framework MESA, ces dimensions se révèlent insuffisantes pour contenir 15 002 agents. Ainsi, certains paramètres tels que *wandering\_speed\_high / wandering\_speed\_low* et *infect\_rangont* dû être ajustés à l'échelle pour un environnement de 700x700.

```
simulation_parameters = {
    "width": 700, "height": 700, "number_S": 15000, "number_I": 2, "p_inf_high": 0.90, "q_inf_high": 0.90, "p_inf_high_after": 0.75, "q_inf_high_after": 0.75,
    "p_inf_low": 0.30, "q_inf_low": 0.30, "gamma": 0.05, "wandering_speed_high": 5.83, "wandering_speed_low": 0.583, "infect_range": 4.664, "max_iterations": 2000,
    "alpha": 10, "coeff_memory": 0.5, "coef_quarantine": 3.0, "amplif_quarantine": 1.1, "atten_quarantine": 0.9, "coef_threshold_in": 0.40, "coef_threshold_out": 0.010,
}
```

Figure 18 – Paramètre du modèle.

Par la suite, le serveur affichera le modèle créé sur une page web en utilisant le module de visualisation correspondant. De plus, à partir des données collectées par le modèle, le serveur générera des graphiques dynamiques. Ce dernier déterminera également la taille, la couleur et la forme des agents présents dans le modèle.

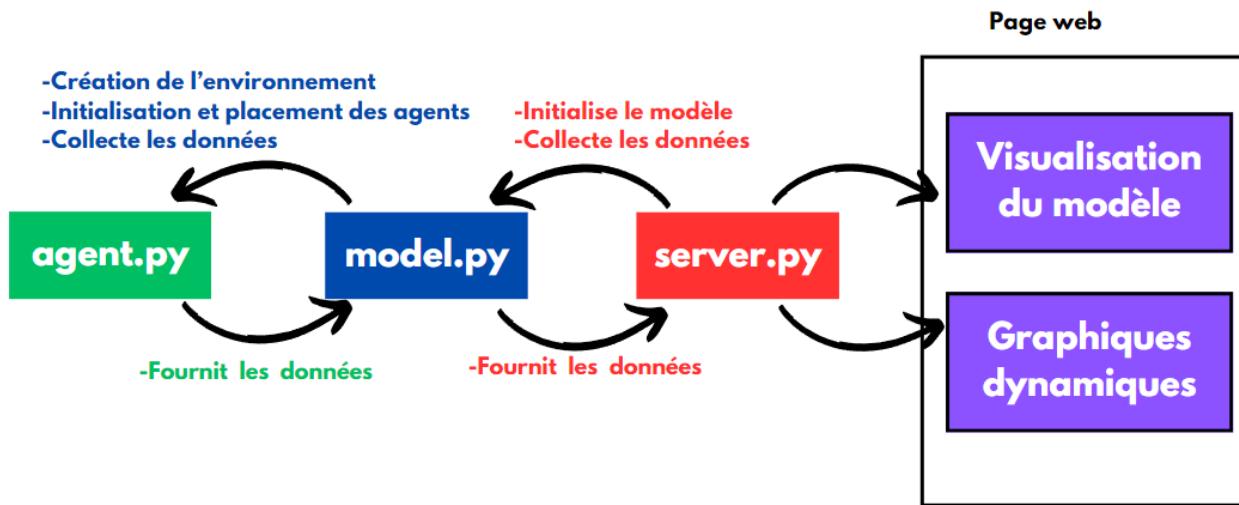


Figure 19 – Rôle du serveur.

Une fois le serveur exécuté, nous trouvons sur la page web, (voir annexe 2 page complète) les éléments ci-dessous :



Figure 20 – Dynamique du taux de variation d'infection.



Figure 21 – Dynamique de l'évolution de la population.

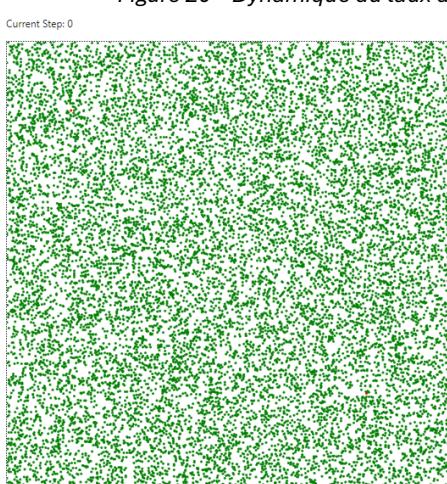


Figure 22 – Environnement avec agent.



Figure 23 – Lancement / arrêt / réinitialisation / changement de vitesse de la simulation

## 2.3 Résultats

Afin d'évaluer l'efficacité du modèle spatial SIR de M. Bădică et de vérifier la fiabilité de mon implémentation, j'ai procédé à une simulation de propagation d'épidémie. Deux scénarios ont été créés : l'un avec un système de contrôle et l'autre sans, comme illustré dans les figures 22 et 23. Les deux simulations partagent des caractéristiques communes telles qu'une taille d'environnement identique ( $700 \times 700$ ), un nombre équivalent d'agents susceptibles (15 000) et un nombre similaire d'agents infectés (2) placés au même emplacement pour chaque simulation. De plus, les paramètres du modèle sont rigoureusement identiques pour les deux scénarios. Il convient également de noter que ces simulations comportent chacune deux foyers distincts d'épidémies initiées par les agents infectés initiaux respectifs (ce qui n'est pas toujours le cas, car il y a une probabilité qu'un agent infecté devienne « retiré » avant de contaminer un agent susceptible). Par ailleurs, pour la simulation avec état d'urgence, celui-ci durera la première fois 60 étapes et variera lors des prochains états d'urgences.

Current Step: 170

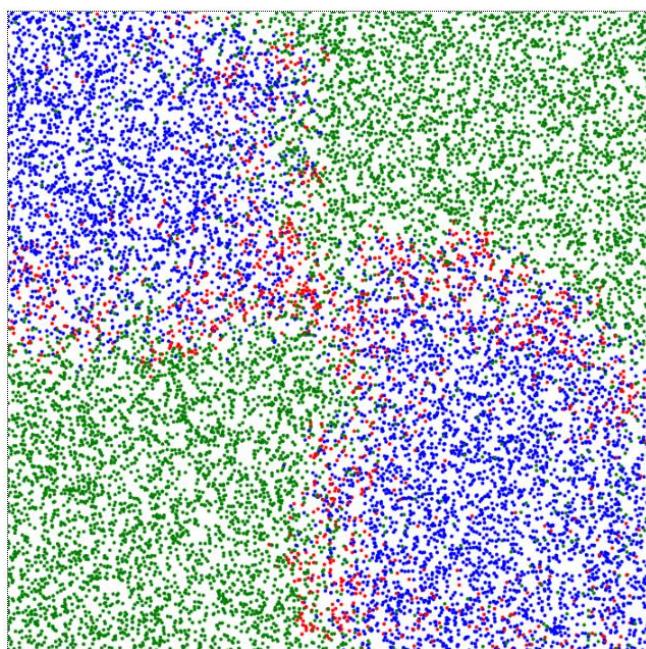


Figure 24.a – Population après 170 cycles.

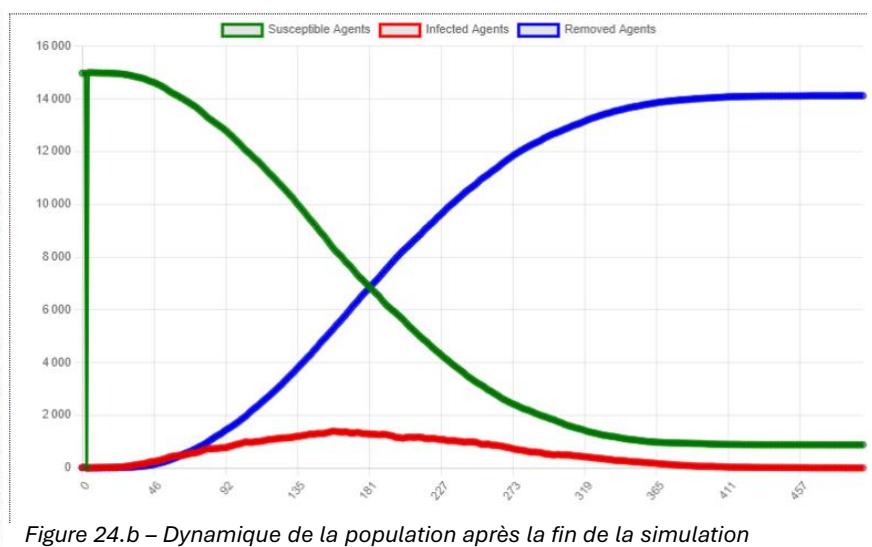


Figure 24.b – Dynamique de la population après la fin de la simulation au cycle 498.

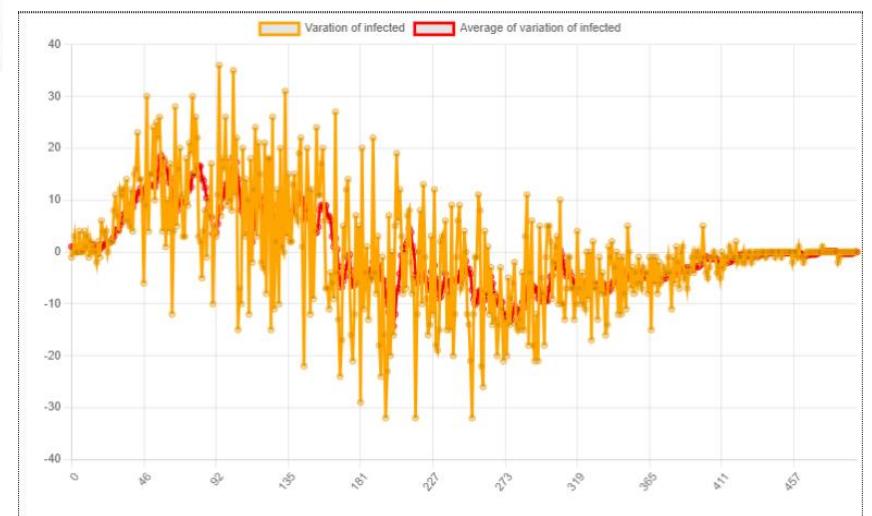


Figure 24.c – Dynamique de taux de variation d'infection.

Current Step: 170

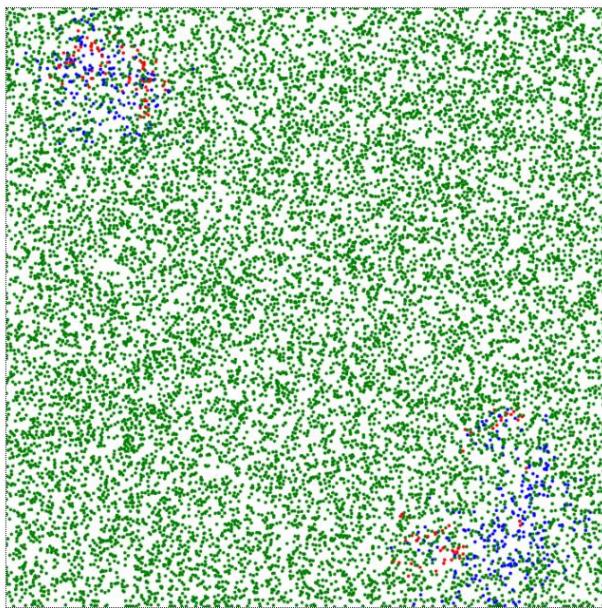


Figure 25.a – Population après 170 cycles.

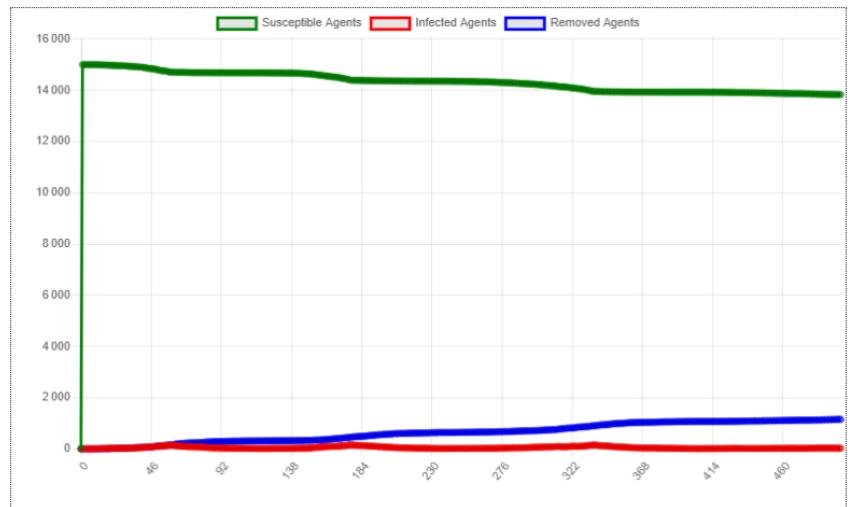


Figure 25.b – Dynamique de la population au cycle 498.

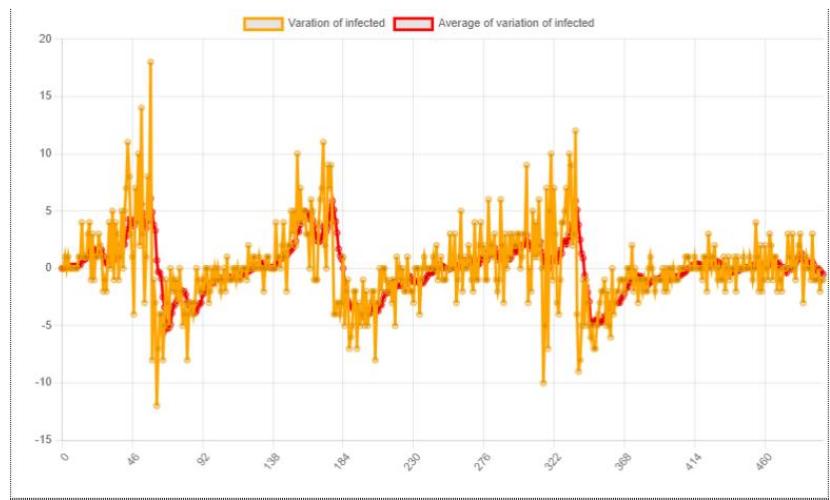


Figure 25.c – Dynamique de taux de variation d'infection.

Il est manifeste que le contraste entre une simulation avec ou sans système de contrôle est clair. Cette observation nous enseigne que, en l'absence de mesures de protection, la propagation d'une épidémie au sein d'une population risque d'affecter la quasi-totalité de cette dernière (Fig.24.b). En revanche, grâce au système de contrôle avec réduction de la vitesse des agents, nous constatons que, d'une part, les propagations de l'épidémie sont ralenties (figure 24.a - 25.a) et que le nombre de personnes touchées par la maladie ne représente que 11,5 % de la population après 498 cycles (figure 25.b), contrairement à la simulation sans système de contrôle où 93% de la population après 498 cycles ont été touché par la maladie. Nous pouvons donc conclure que limiter les déplacements des personnes s'avère une solution efficace pour gérer la propagation d'une épidémie, comme cela a été observé dans cette simulation ainsi qu'au cours des événements récents tels que celui du Covid-19. Néanmoins, sur la figure 25.a, on note deux foyers épidémiques. Dès lors se pose la question : est-il nécessaire de placer tous les individus en quarantaine même s'ils se trouvent loin de ces foyers ? Car bien que la limitation des déplacements empêche une propagation, elle prive également les individus de leur liberté.

### 3 Ajout de nouvelles stratégies

Pour chacun des divers ABMS que je vais exposer, ils se fondent tous sur ce que j'ai présenté précédemment avec le même système de contrôle. Ce qui va différer, c'est l'implémentation de la procédure de protection et/ou l'adoption d'une nouvelle mesure de protection en cas d'état d'urgence. Je présenterai les résultats obtenus de chaque ABMS avec un contexte identique : taille d'environnement (700x700), un nombre équivalent d'agents susceptibles (15 000) et un nombre similaire d'agents infectés (2) placés au même emplacement pour chaque simulation. De plus, les paramètres du modèle sont rigoureusement. Il convient également de noter que chaque simulation comporte deux foyers distincts d'épidémies initiées par les agents infectés initiaux respectifs et qu'à la première itération d'un état d'urgence, celui-ci durera 60 étapes, puis variera lors des prochaines itérations en fonction des paramètres suivant : *coef\_quarantine/amplif\_quarantine/atten\_quarantine*. L'objectif de réaliser ces simulations dans un contexte identiques et de comparer l'efficacité de chaque ABMS.

#### 3.1 Système de création de zones statiques

##### 3.1.1 Développement

Une première réponse à la question posée et la création de région statique pour diminuer le nombre d'agents qui auront leur vitesse réduite durant un état d'urgence. Ce dispositif divise l'environnement du modèle en quatre parties distinctes. Lorsque le modèle signale une situation d'urgence, il examine chaque zone statique en vérifiant si la variation du nombre d'agents infectés dépasse le seuil requis pour déclencher un état d'urgence :  $\frac{\text{threshold\_in}}{4}$  (étant donné les quatre régions). Si tel est le cas pour certaines régions spécifiques, tous les agents se trouvant dans ces zones verront leur vitesse grandement diminuée.

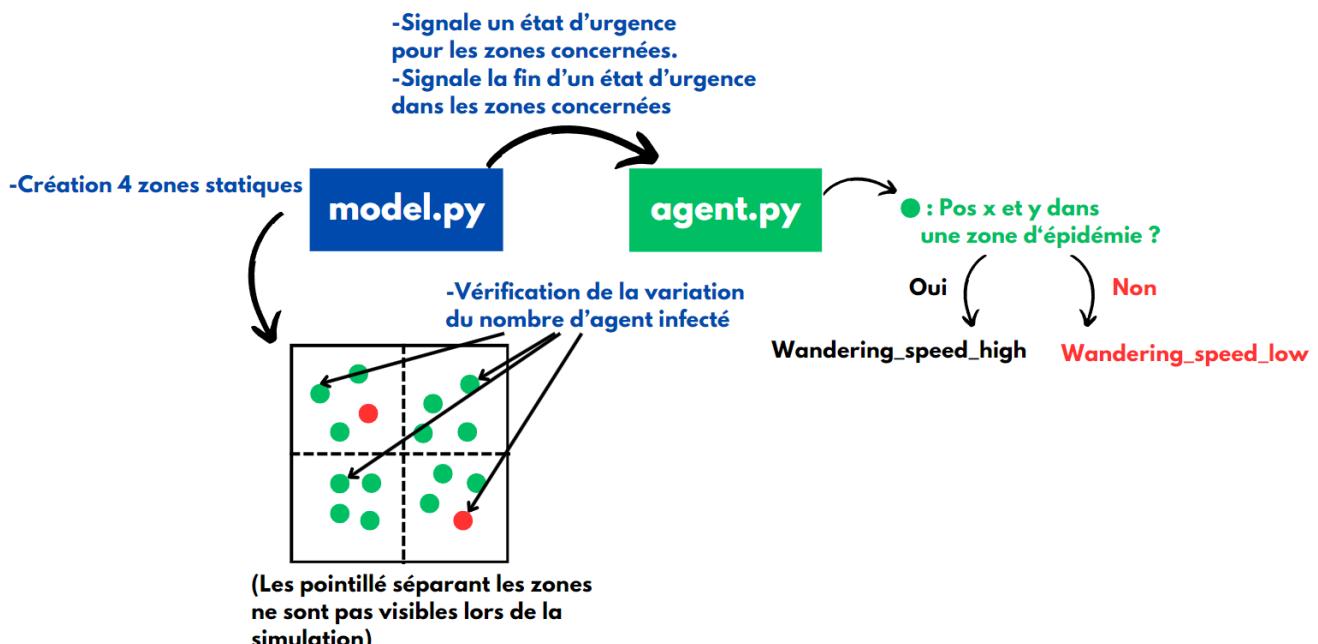


Figure 26– Fonctionnement de système à région statique.

### 3.1.2 Résultat

Current Step: 174

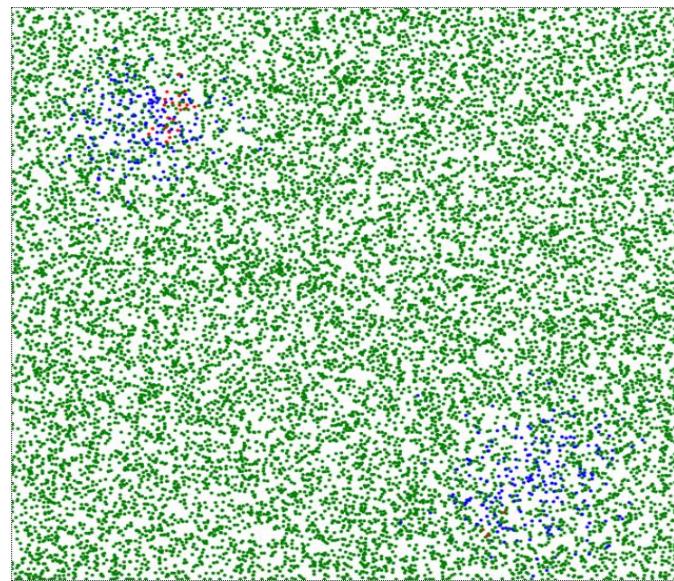


Figure 27.a – Population après 170 cycles.

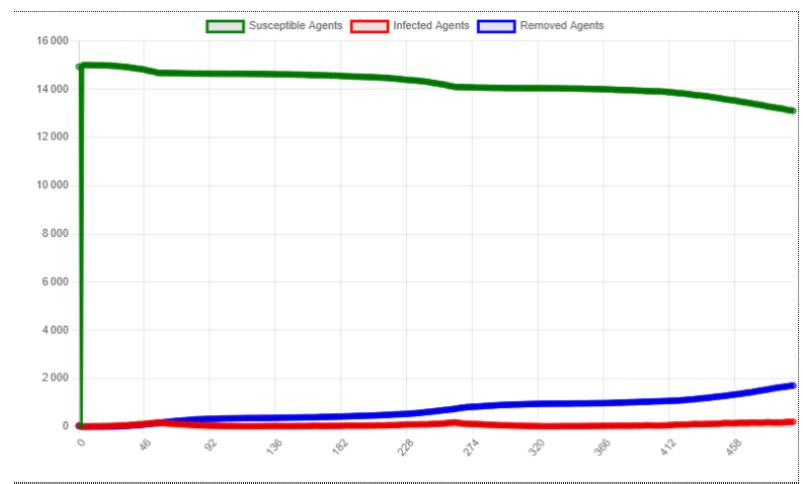


Figure 27.b – Dynamique de la population après la fin de la simulation au cycle 498.

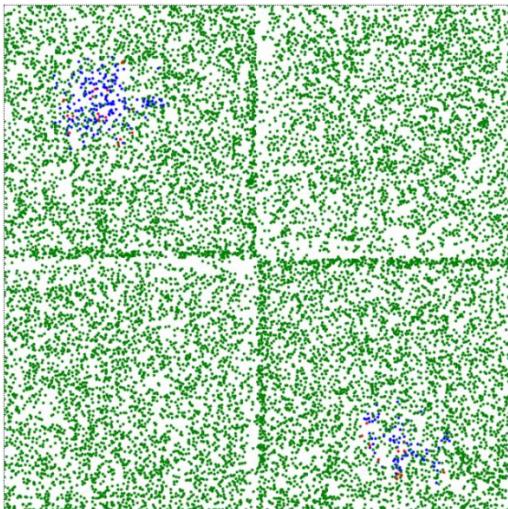


Figure 27.d – Exemple zone statique lors de l'état d'urgence. (Haute gauche et bas droite)

Il est à noter que les résultats de la simulation sont remarquablement similaires à ceux du premier système, même si une partie de la population d'agents voit sa vitesse considérablement réduite. Ainsi, il est possible de conclure qu'en cas d'épidémie, l'application de mesures de protection à l'ensemble de la population n'est pas nécessaire.

Cependant, il convient d'explorer la possibilité de réduire le nombre d'agents affectés afin de garantir un maximum de liberté aux autres agents. En effet, le principal enjeu de ce système réside dans le fait que, même en cas d'épidémie dans une zone spécifique, tous les agents de ladite zone pourraient ne pas être directement touchés en raison de leur éloignement. Par conséquent, l'application de mesures préventives à leur égard serait inefficace.

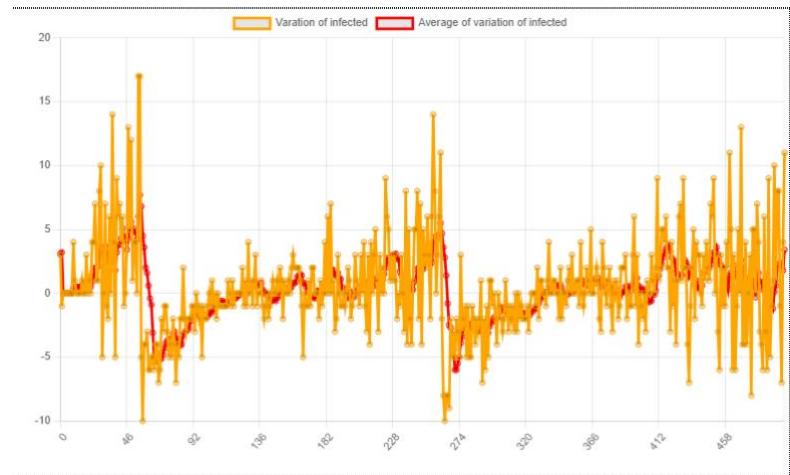
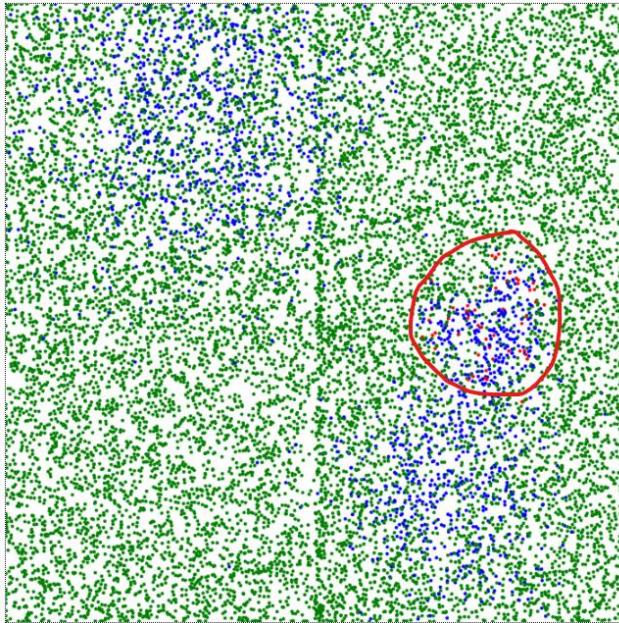


Figure 27.c – Dynamique de taux de variation d'infection.



On constate la présence d'agents infectés (représentés en rouge) situés au bas de la région supérieure droite. Si ce nombre dépasse le seuil requis pour déclencher une mise en quarantaine, l'ensemble de la région sera impacté, alors que les agents localisés plus haut dans cette même région se trouvent loin de l'épidémie et ne sont donc pas concernés.

Figure 28-Exemple du problème des régions statiques.

### 3.2 Système de création de zones dynamiques

Le système de régions dynamique des épidémies ne possède aucune région prédéfinie à l'avance par le modèle. Les régions sont déterminées en fonction des foyers d'épidémies lorsque le modèle détecte une situation d'urgence. Ainsi, il est possible qu'il y ait plus de quatre zones placées en quarantaine, mais moins d'agents seront verront leurs vitesses réduites.

Afin d'accomplir cette tâche, j'ai entrepris une recherche approfondie pour dénicher des solutions permettant de créer des zones dynamiques basées sur l'évolution de la simulation. C'est ainsi que j'ai découvert les algorithmes de regroupement, communément appelés « clustering ». Ces derniers constituent une technique d'analyse de données largement utilisée en statistiques et en apprentissage automatique non supervisé. L'apprentissage non supervisé représente une méthode fondamentale dans le domaine du machine learning, où le modèle est entraîné sur des données non étiquetées. Contrairement à l'apprentissage supervisé, qui repose sur un ensemble de données d'entraînement comprenant à la fois des entrées et des sorties correspondantes (étiquettes), l'apprentissage non supervisé se contente uniquement des entrées sans les sorties associées.

L'objectif principal de l'apprentissage non supervisé est de découvrir les structures cachées dans les données. Ainsi, les agents (qui peuvent être infectés, susceptibles ou immunisés) se déplacent dans un espace continu. Ces agents ne possèdent pas de groupe préalablement attribué (cluster). Leurs positions sont simplement représentées par des coordonnées ( $x, y$ ) dans l'espace simulé, constituant ainsi des données non étiquetées.

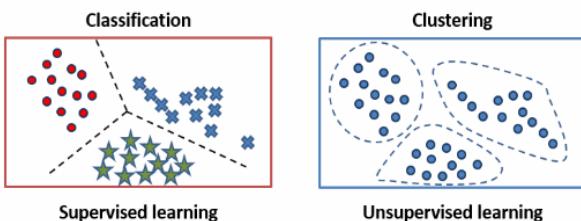


Figure 29-Supervisée / non supervisée.

Le but principal du clustering réside dans la segmentation d'un ensemble de données en groupes homogènes, appelés clusters, où les éléments au sein d'un même groupe présentent des similitudes. Cette approche me permettra de délimiter des zones épidémiques regroupant principalement des agents infectés et « retirés ». Après une analyse approfondie des différents algorithmes, j'ai opté pour l'algorithme de *K – Means*.

### 3.2.1 *K – Means*

L'algorithme *K – Means* est une méthode de regroupement utilisée en analyse de données pour diviser un ensemble de données en  $k$  groupes distincts. Chaque groupe est formé de points de données qui sont proches les uns des autres, selon une certaine mesure de distance. L'objectif principal de l'algorithme est de minimiser la distance totale entre chaque point de donnée et le centre de son groupe (appelé centroïde).

L'algorithme de *Lloyd*, également connu sous le nom de procédure de *Lloyd*, ou d'algorithme de *Lloyd*, est une méthode itérative couramment utilisée pour implémenter l'algorithme *K – Means*.

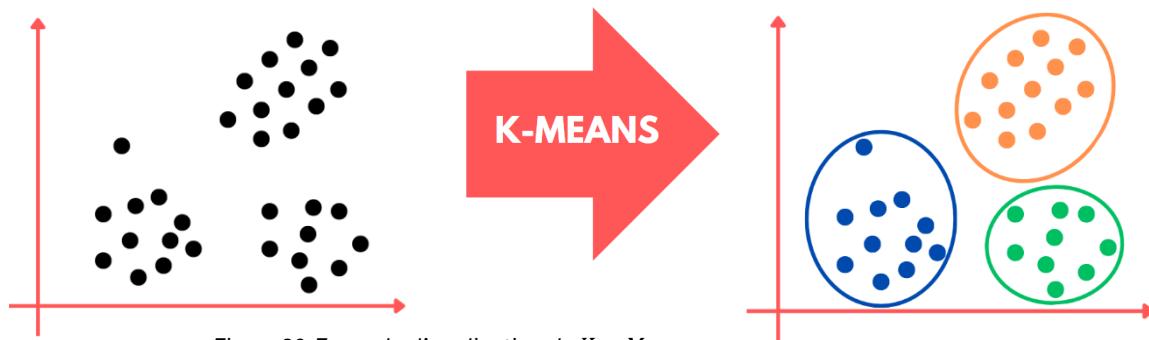


Figure 30-Exemple d'application de *K – Means*.

### 3.2.1.2 Algorithme de *Lloyd*

L'algorithme de *Lloyd* est une heuristique de recherche locale itérative qui fonctionne sur des points d'un espace euclidien  $\mathbb{R}^d$ . La mesure qu'il cherche implicitement à optimiser est la fonction de coût *k – means*. Pour un ensemble de points  $X \subset \mathbb{R}^d$  et un ensemble de centres  $C \subset \mathbb{R}^d$ , la fonction de coût k-means est définie comme suit :

$$\Phi(X, C) = \sum_{x \in X} \min_{c \in C} \|x - c\|^2$$

- $X$  : L'ensemble de tous les points de données dans l'espace euclidien  $\mathbb{R}^d$ .

- $C$  : L'ensemble des centroïdes des clusters, où  $|C| = k$ , et  $k$  est le nombre de clusters à trouver.

$\|x - c\|^2$  : La distance euclidienne au carré entre un point de données  $x$  et un centroïde  $c$ .

$\min_{c \in C} \|x - c\|^2$  : La distance euclidienne au carré minimale entre un point de données  $x$  et n'importe quel centroïde  $c$  dans l'ensemble  $C$ .

$\sum_{x \in X} \square$  : La somme de toutes les distances euclidiennes au carré minimales entre chaque point de données  $x$  dans l'ensemble  $X$  et leurs centroïdes respectifs dans l'ensemble  $C$ .

En somme, la fonction de coût  $\Phi(X, C)$  mesure la somme des distances euclidiennes au carré minimales entre tous les points de données et leurs centroïdes de cluster respectif. L'objectif de l'algorithme de *Lloyd* est de minimiser cette fonction de coût en itérant entre l'étape d'attribution (assigner les points de données aux clusters les plus proches) et l'étape de mise à jour (recalculer les centroïdes des clusters).

### 3.2.1.3 Fonctionnement de *K – Means*

**1. Initialisation :** sélection aléatoire  $k$  points de données en tant que centroïdes initiaux.

**2. Attribution** de chaque point de donnée au cluster dont le centroïde est le plus proche, en utilisant une mesure de distance euclidienne.

**3. Mise à jour** (Algorithme de *Lloyd*) : recalcul des nouveaux centroïdes pour chaque cluster. Cela se fait en prenant la moyenne de tous les points de données assignés à chaque cluster. Après avoir mis à jour les centroïdes, on répète les étapes d'attribution et de mise à jour jusqu'à ce que les centroïdes convergent ou qu'un critère d'arrêt prédéfini soit atteint.

**4. Évaluation** : on évalue si les centroïdes ont convergé ou si un critère d'arrêt prédéfini est atteint. Si ce n'est pas le cas, on retourne à l'étape d'attribution et continue le processus jusqu'à obtenir la convergence ou atteindre le critère d'arrêt.

#### Illustration

**Etape 1 :** Sélection du nombre  $k$  de clusters que l'on veut identifier dans nos données.  $k = 3$ . 3 points distincts choisis au hasard parmi ceux disponibles.

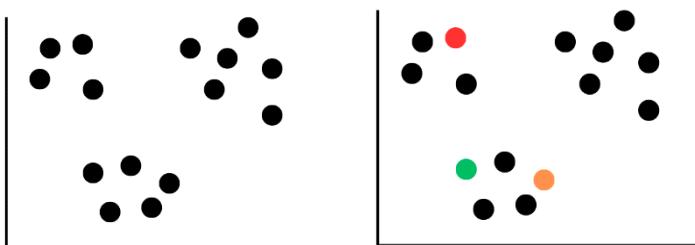


Figure 31-Illustration *K – Means* 1.

**Etape 2 :** Attribution de chaque point au cluster le plus proche avec distance euclidienne.

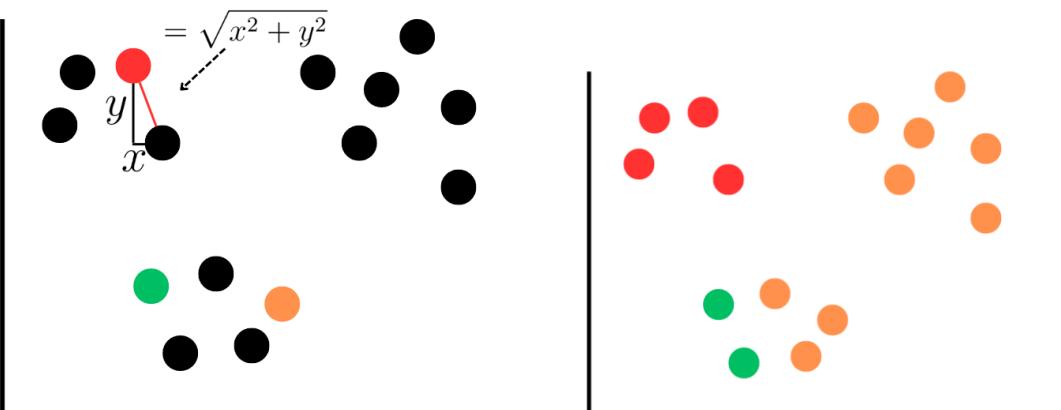
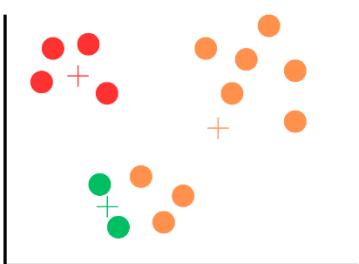


Figure 32-Illustration *K – Means* 2.

**Etape 3:** Calcule de la moyenne, «*mean*», de chaque cluster.



Puis, nous répétons l'étape deux jusqu'à ce que les centroïdes ne bougent plus.

Figure 33-Illustration *K – Means* 3.

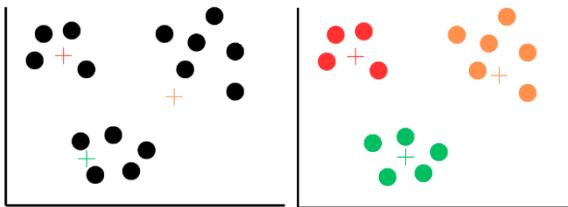


Figure 34-Illustration K – Means 4.

**Etape 4 :** Evaluation s'il y a eu convergence ou nombre d'itérations atteintes.

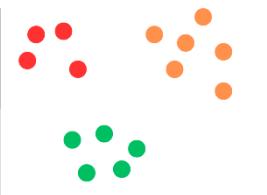


Figure 35-Illustration K – Means 5.

**Remarque :** J'ai choisi l'algorithme *K – Means* car il est populaire pour sa simplicité et son efficacité sur des ensembles de données de grande taille. En effet, j'ai essayé d'utiliser un autre algorithme de clustering : *DBSCAN* mais celui-ci est d'une complexité supérieure ce qui réduit la vitesse d'exécution de la simulation avec de nombreux agents.

### 3.2.2 *K – Means* ++

Pour mon système de région d'épidémie dynamique, j'ai utilisé la librairie python *scikit – learn*. Cependant, celle-ci implémente une variante de l'algorithme de *K – Means* : le *K – Means* ++ qui est une méthode d'initialisation intelligente des centroïdes pour l'algorithme *K – Means*. L'objectif est de répartir les centroïdes initiaux en attribuant le premier centroïde de manière aléatoire, puis en sélectionnant le reste des centroïdes en fonction de la distance au carré maximale. L'idée est de pousser les centroïdes le plus loin possible les uns des autres.

En d'autres termes, *K – Means* ++ cherche à améliorer l'initialisation des centroïdes dans l'algorithme *K – Means* en choisissant des points de départ plus éloignés les uns des autres, ce qui peut conduire à une convergence plus rapide et à de meilleurs résultats finaux. Cette méthode est particulièrement utile lorsque les données sont mal réparties ou ont des structures complexes.

#### 3.2.2.1 Fonctionnement de *K – Means* ++

1. Sélection aléatoire du premier centroïde depuis les données.
2. Pour chaque données  $x$ , calcule de  $d(x)$ , où  $d(x)$  est la distance minimum de  $x$  jusqu'à un centroïde.
3. Puis vient le choix du prochain centroïde depuis les données avec la probabilité qu'une donnée  $x$  devienne un centroïde proportionnellement à  $d(x)^2$ .
4. Et on répète les étapes 2 et 3 jusqu'à obtenir le nombre de centroïdes souhaité.
5. Une fois que les centroïdes initiaux sont sélectionnés à l'aide de l'algorithme *K – Means* ++, l'algorithme *K – means* classique est utilisé pour affecter les points aux clusters et mettre à jour les centroïdes jusqu'à ce que la convergence soit atteinte.

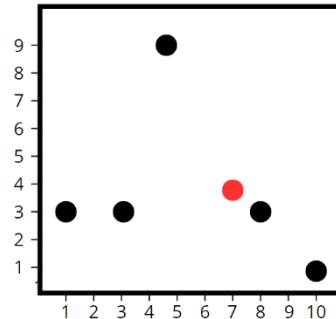
#### Illustration

Admettons que nous avons 6 points : [(7,4), (8,3), (5,9), (3,3), (1,3), (10,1)]  
Nous voulons 3 clusters.

**Etape 1**

Nous choisissons au hasard comme premier centroïde : (7,4)

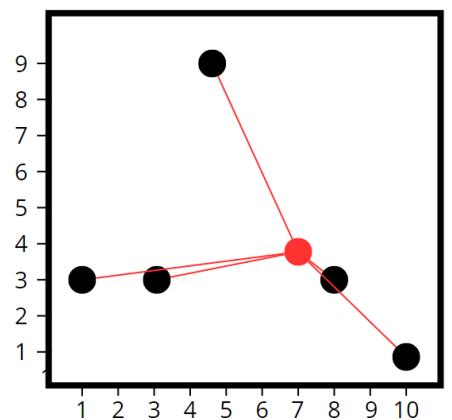
Figure 36-Illustration K – Means ++ 1.



**Etape 2:** Calcule de la distance minimum entre le centroïde et les autres points.

$x$	$\min(d(x, z_i)^2)$
(7, 4)	—
(8, 3)	2
(5, 9)	29
(3, 3)	17
(1, 3)	37
(10, 1)	18

Figure 37-Illustration K – Means ++ 2.



**Etape 3 :** sélection du prochain centroïde en fonction des probabilités. (Le plus loin = plus de chance).

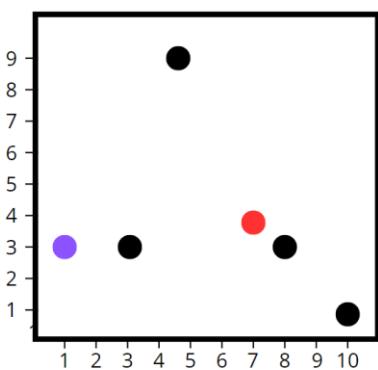
$x$	prob
(7, 4)	—
(8, 3)	2/103
(5, 9)	29/103
(3, 3)	17/103
(1, 3)	37/103
(10, 1)	18/103

Ainsi c'est le point (1,3) qui a la plus grande probabilité de devenir le prochain centroïde.

Figure 38-Illustration K – Means ++ 3.

**Etape 4:** Nous itérons une nouvelles fois les étapes 2 et 3 avec cette fois-ci 2 centroïdes :

**Etape 2 – 2<sup>ème</sup> itération :**



$x$	$\min(d(x, z_i)^2)$
(7, 4)	—
(8, 3)	2
(5, 9)	29
(3, 3)	4
(1, 3)	—
(10, 1)	18

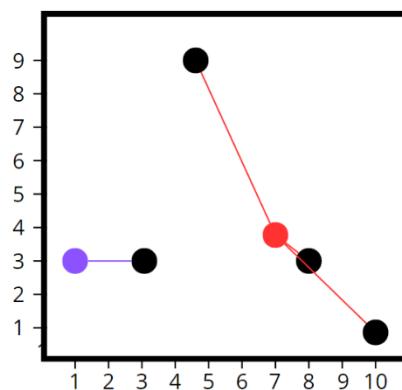


Figure 39 - Illustration K – Means ++ 4.

**Etape 3 – 2<sup>ième</sup> itération :**

$x$	prob
(7, 4)	—
(8, 3)	2/55
(5, 9)	29/55
(3, 3)	4/55
(1, 3)	—
(10, 1)	18/55

Ainsi celui qui a le plus de chance d'être le prochain centroïde est le point (5,9).

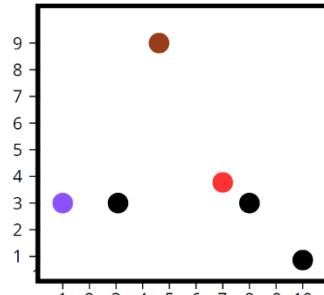


Figure 40 - Illustration K – Means + + 5.

**Etape 5 :** Nous pouvons maintenant débuter le *K – means* avec les centroïdes initié.

Lorsque la taille du jeu de données est importante, nous n'avons pas forcément d'hypothèses sur les données, et par la suite, nous ne pouvons pas choisir la valeur de  $k$  clusters intuitivement. Comment faire dans ce cas ?

### 3.2.3 Score silhouette

Il existe une méthode empirique qui nous permet de déterminer une valeur de  $k$  optimale lorsque l'on ne sait pas exactement combien de clusters il existerait dans le jeu de données : La méthode de silhouette.

Le coefficient de silhouette est une mesure de la similitude d'un point de données à l'intérieur d'un groupe par rapport à d'autres groupes. Ce coefficient doit être calculé pour chaque point du jeu de données. Pour trouver ce coefficient du  $i$  – ème point :

- On commence par calculer  $a_i$  : la distance moyenne de ce point avec tous les autres points dans les mêmes clusters.
- Ensuite, on calcule  $b_i$  : la distance moyenne de ce point avec tous les points du cluster le plus proche de son cluster.
- Finalement, on calcule  $s_i$  : (coefficient de silhouette) pour cet  $i$ -ème point en utilisant la formule ci-contre :  $s_i = \frac{b_i - a_i}{\max(b_i - a_i)}$

On choisira une plage de valeurs  $k$  candidates, puis on lancera l'algorithme *K – Means* pour ces valeurs de  $k$  en calculant à chaque fois le coefficient de silhouette moyen.

**Remarque :** d'autres méthodes sont disponibles pour déterminer une valeur de  $k$  cluster optimale, telles que la méthode du coude (Elbow), que j'ai tenté de mettre en œuvre. Cependant, les résultats obtenus ne correspondaient pas toujours à mes attentes, car le nombre de clusters n'était pas optimal contrairement au score silhouette.

### 3.2.4 Développement

Pour pouvoir implémenter un système de région dynamiques dans le modèle j'ai utilisé l'algorithme de  $K - Means$  ++ de la librairie python *sckit - learn* et la métrique de silhouette score.

```
def update_epidemic_zones(self) -> None:
    #---PART1---
    infected_positions = [agent.pos for agent in self.schedule.agents if agent.is_infected]
    if infected_positions:
        if self.epidemic_zones == []:
            #---PART2---
            optimal_clusters = 0
            max_silhouette_score = -1

            for k in range(2, len(infected_positions) // 2 + 1):
                kmeans = KMeans(n_clusters=k)
                kmeans.fit(infected_positions)
                cluster_labels = kmeans.labels_
                silhouette_score_ = silhouette_score(infected_positions, cluster_labels) #10

                if silhouette_score_ > max_silhouette_score:
                    max_silhouette_score = silhouette_score_
                    optimal_clusters = k
            #---PART3---
            kmeans = KMeans(n_clusters=optimal_clusters)
            clustered_positions = kmeans.fit_predict(infected_positions)
            for cluster in set(clustered_positions):

                cluster_positions = [pos for pos, clust in zip(infected_positions, clustered_positions) if clust == cluster] #17
                avg_distance = np.mean([np.linalg.norm(a - b) for a, b in itertools.combinations(cluster_positions, r=2)]) #18

                center = np.mean(cluster_positions, axis=0) #19

                radius = avg_distance * 1.5

                self.epidemic_zones.append((center, radius))

            self.epidemic_zones_updated = True
```

Figure 41 – Algorithme de la création de régions dynamique d'épidémie.

**Première partie de la méthode :** la méthode commence par récupérer les positions des agents infectés dans l'espace continu en utilisant une liste en compréhension. Si la liste n'est pas vide, c'est-à-dire qu'il y a au moins un agent infecté.

**Deuxième partie de la méthode :** la méthode itère sur les valeurs de  $k$  allant de 2, car il faut minimum 2 clusters pour l'algorithme  $k - Means$  ++, à la moitié de la longueur de *infected\_position*, (nombre d'agents infecté), arrondie à l'entier supérieur pour tester un nombre raisonnable de valeurs de  $k$  tout en évitant de tester des valeurs trop élevées qui pourraient prendre beaucoup de temps à calculer et ne pas donner de meilleurs résultats.

Ensuite elle initialise le modèle de clustering  $K - Means$  avec un nombre de clusters égal à  $k$ . Puis, avec la méthode *fit*, l'algorithme  $k - means$  ++ se lance en ne prenant en compte que les agents infectés. *kmeans.labels\_* est un attribut de l'objet *KMeans* de la bibliothèque *scikit - learn*, qui est généré après l'appel de la méthode *fit*. Cet attribut est un tableau de taille *n\_samples* (le nombre d'échantillons dans les données d'entrée), contenant les étiquettes de cluster assignées à chaque échantillon par l'algorithme de clustering  $K - Means$ .

Chaque valeur dans `kmeans.labels_` correspond à l'indice du cluster auquel l'agent a été assigné, avec des valeurs allant de 0 à  $k - 1$  (où  $k$  est le nombre de clusters spécifié dans l'initialisation de l'objet `KMeans`). Ce qui permet de savoir à quel cluster appartient chaque agent infecté.

Avec la ligne #10, la méthode calcule le score de silhouette des clusters obtenus après l'application de l'algorithme *K – Means++*. Le score de silhouette varie entre  $-1$  et  $1$ . Un score proche de  $1$  indique que le point est bien assigné à son cluster et est bien séparé des autres clusters. Un score proche de  $-1$  indique que le point est mal assigné à son cluster et est plus proche des autres clusters. Un score proche de  $0$  indique que le point se trouve à la frontière entre deux clusters. A la fin des itérations nous obtenons le nombre optimal de clusters grâce à ce score.

**Dernière partie de la méthode :** une fois le nombre optimal de clusters déterminé, la méthode initialise le modèle de clustering *K – Means* avec le nombre optimal de clusters. La méthode `fit_predict` est utilisée pour entraîner le modèle *KMeans* sur les données d'entrée, les agents infectés, et prédire les étiquettes de cluster pour chaque point de données en une seule étape. Elle renvoie un tableau de labels, où chaque label correspond à l'étiquette de cluster prédictive pour chaque point de données, stockées dans `clustered_positions`.

Remarque : on utilise `set(clustered_positions)` pour itérer car c'est un tableau de labels, où chaque valeur représente l'identifiant du cluster auquel appartient un point de données. En utilisant `set`, on crée un ensemble unique de tous les identifiants de cluster présents dans `clustered_positions`. Cela permet d'itérer sur chaque cluster unique présent dans les données, plutôt que sur chaque point de données individuellement, car cela pourrait entraîner des calculs redondants et inutiles étant donné que plusieurs points de données peuvent appartenir au même cluster.

A la ligne #17, l'utilisation de `zip(infected_positions, clustered_positions)` permet de combiner deux listes en une seule séquence de paires (tuples). Le résultat sera une liste de tuples où chaque tuple contient une position d'agent infecté et le label de cluster correspondant. Cette ligne sert à extraire les positions des agents infectés qui appartiennent à un cluster spécifique.

La ligne #18 permet de calculer la distance moyenne entre toutes les paires d'agents infectés présentes dans le cluster en cours de traitement. Pour ce faire, elle recourt à la fonction `itertools.combinations()` afin de générer l'ensemble des paires possibles d'agents infectés au sein du cluster, sans duplication. Ensuite, pour chaque paire d'agents infecté  $a$  et  $b$ , elle évalue la distance euclidienne les séparant en utilisant la fonction `np.linalg.norm(a - b)`. Enfin, elle établit la moyenne de toutes ces distances grâce à l'utilisation de la fonction `np.mean()`.

La ligne #19 est utilisée pour calculer le centre du cluster actuellement traité en prenant la moyenne de toutes les positions dans ce cluster.

La méthode est désormais en mesure de calculer le rayon de la zone de quarantaine pour le cluster actuellement pris en charge, en multipliant la distance moyenne entre les points du cluster (`avg_distance`) par un coefficient d'agrandissement, fixé à  $1,5$  dans ce cas spécifique. Ce choix a été fait de manière arbitraire afin d'établir une zone de quarantaine suffisamment étendue pour englober tous les agents infectés du cluster, tout en évitant qu'elle ne soit trop grande et n'inclue des agents situés trop loin du cluster. Ainsi, la zone de quarantaine sera circulaire, avec comme centre

le centre du cluster (calculé précédemment), et son rayon sera égal à 1,5 fois la distance moyenne entre les points du cluster.

Les localisations des centres ainsi que les rayons sont intégrées à la liste des zones épidémiques pour chaque cluster.

En dernier lieu, la méthode attribue la variable `epidemic_zones_updated` à `True` afin d'indiquer que les zones d'épidémies ont été mises à jour, évitant ainsi l'exécution répétée de cette méthode à chaque étape de la simulation durant l'état d'urgence. En effet, l'algorithme  $K - Means +$  présente une complexité en  $O(lnk)$  et nécessite plusieurs exécutions pour déterminer initialement le nombre optimal  $k$  de clusters grâce au score silhouette, avant de le réexécuter avec le bon nombre de clusters. L'avantage potentiel d'une exécution continue de cette méthode à chaque étape pendant l'état d'urgence aurait été la réduction de la taille des zones mises en quarantaine pour libérer davantage d'agents si le nombre d'agents infectés diminue au cours de ladite période critique, entraînant ainsi une diminution proportionnelle de la taille des zones concernées.

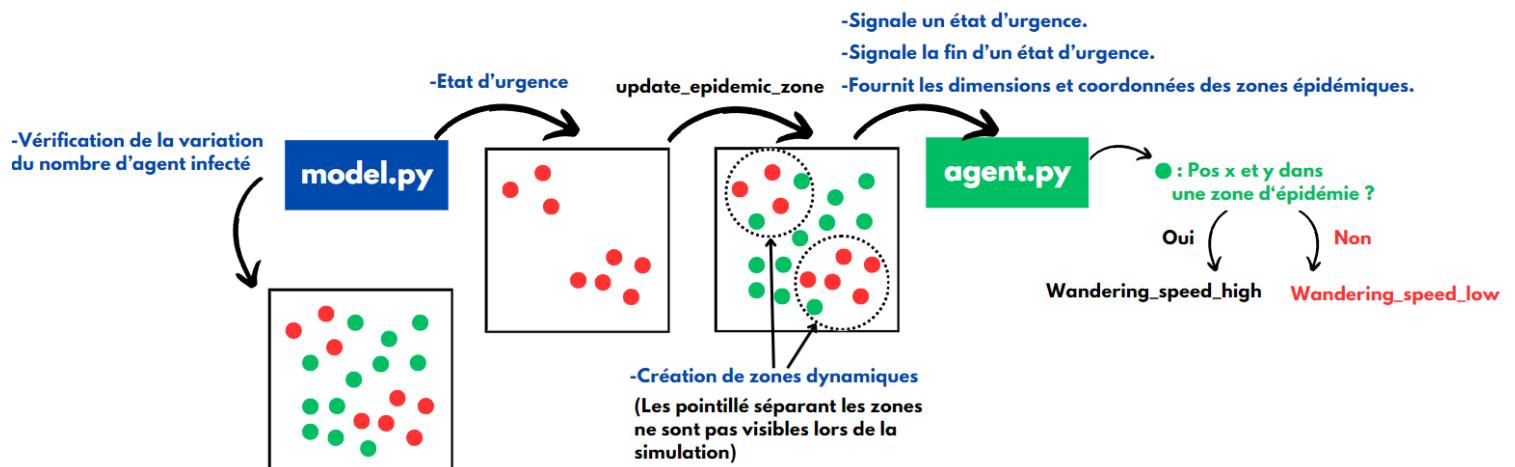


Figure 42 – Fonctionnement du système à région dynamique.

### 3.2.5 Résultat

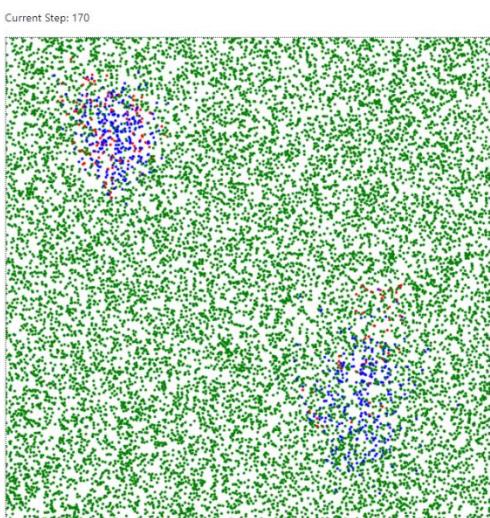


Figure 43.a- Population après 170 cycles.

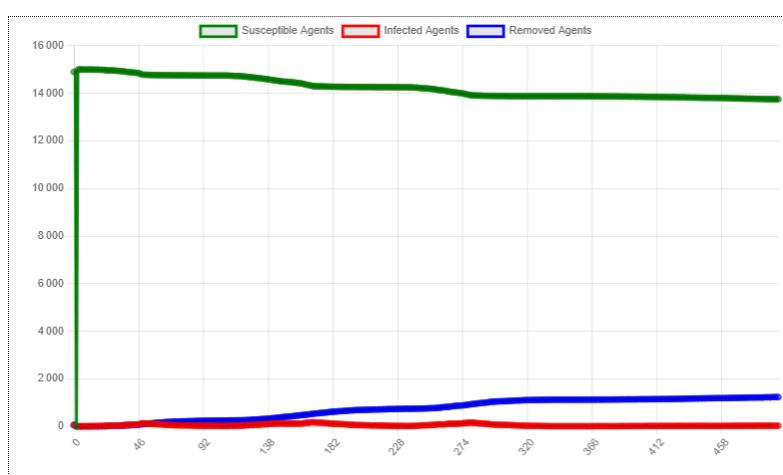


Figure 43.b- Dynamique de la population après la fin de la simulation au cycle 498.

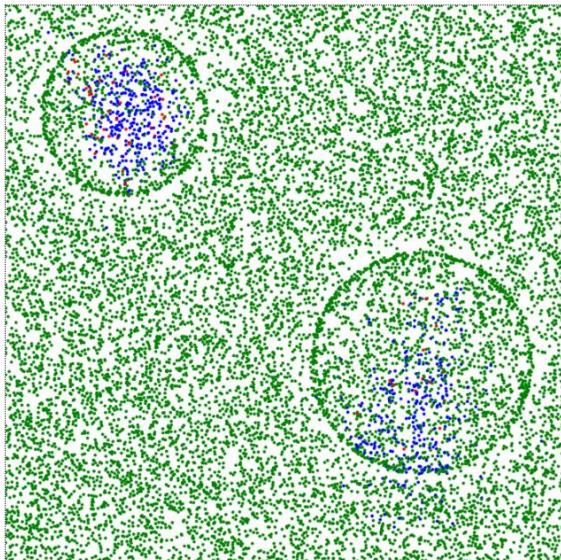


Figure 43.d – Exemple zone dynamique lors de l'état d'urgence.

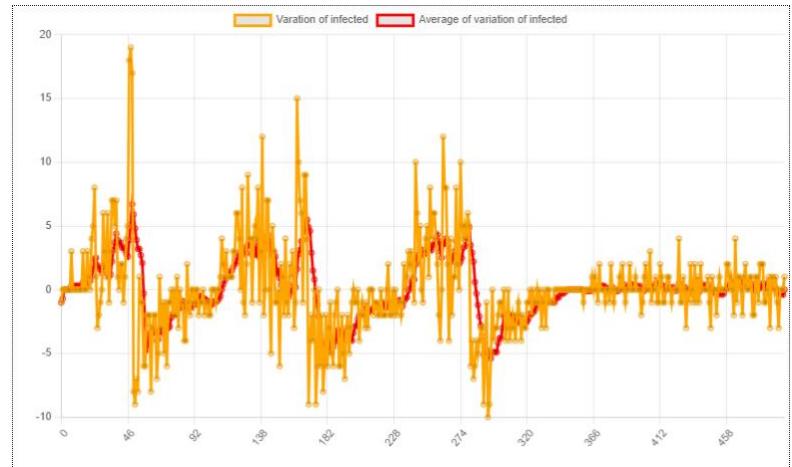


Figure 43.c – Dynamique de taux de variation d'infection.

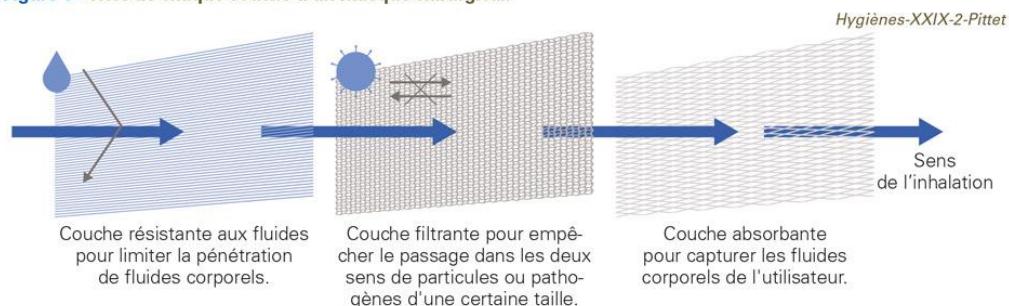
Il est à noter que les résultats obtenus présentent une grande similitude avec ceux des ABMS précédents. En effet, bien que le nombre d'individus qui se voient limité dans ses mouvements soit inférieur par rapport aux deux autres systèmes, les résultats demeurent pratiquement identiques. Ainsi, il est possible de conclure qu'en période de pandémie, il s'avère efficace de cibler les foyers d'épidémies et d'établir un périmètre pour mettre en place des mesures de protection, en veillant à ce que celui-ci ne soit ni trop vaste, afin de ne pas inclure des individus éloignés des foyers infectieux, ni trop restreint pour regrouper dans une même zone tous les individus atteints par la maladie.

La réduction considérable du mouvement d'un agent représente une solution hautement efficace contre la propagation d'une épidémie ; cependant, il restreint la liberté individuelle et exerce un impact sur le monde du travail et l'économie nationale.

### 3.3 Système de port du masque

Une autre solution, qui a été appliquée durant la pandémie du COVID 19 est le port du masque. Nous avons tendance à penser que les masques sont à sens unique et qu'il protège que le porteur. En effet, les masques protègent dans les deux directions lors de l'inspiration et de l'expiration.

Figure 1 – Rôle de chaque couche d'un masque chirurgical.



Adapté de : Chua MH, Cheng W, Goh SS, et al. Face masks in the new Covid-19 normal: materials, testing, and perspectives. Research (Wash D C) 2020;2020:7286735 [1].

Figure 44- Fonctionnement d'un masque.

Ainsi, si deux individus portent un masque avec une efficacité de 50%, le premier masque réduit la probabilité de transmission de 50%, et le second masque également ce qui amène à une réduction de 75%, : $(100 - (100 * 0.5) * 0.5)$ , avec des masques initiaux efficace à 50%,. Ce qui montre l'efficacité indéniables des masques.

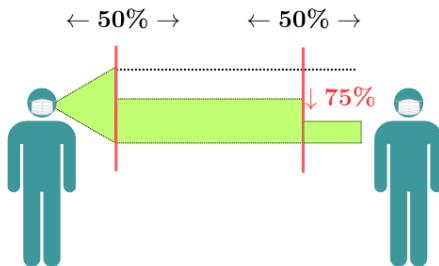


Figure 45- Exemple efficacité du masque avec 50%.

En effet, de nombreux types de masques existent allant du masque en tissu jusqu'au masque chirurgicaux simple ou de type FFP2/KN95/N95 et ils réduisent tous la probabilité de transmission et de contraction de la maladie. A travers de nombreux sites concernant la réduction de ces probabilités, il est ressorti d'après <https://theconversation.com/utilis-ou-non-le-point-sur-l-efficacite-des-masques-contre-le-covid-199548>, qu'en moyenne cette réduction varie de 50 à 80%. Ainsi, pour mon modèle cette probabilité d'atténuation sera de 65%.

Ainsi, si tous les individus portent un masque, la probabilité va être réduite une première fois de 65%, expiration, puis une deuxième fois de 65%, inhalation, ce qui amène à un taux de réduction de 87,8% :  $(100 - (100 * 0.65) * 0.65)$ .

Cependant, pour refléter la réalité tous les agents ne porteront pas de masque. Ainsi, 4 cas de figures sont envisageables pour la simulation. 1) aucun des deux agents ne portent un masque. 2) Seulement l'agent infecté porte un masque. 3) seule l'agent susceptible porte un masque. 4) l'agent susceptible et l'agent infecté portent un masque. Dans ma simulation, les agents « retirés » pourront également porter un masque, car pendant la pandémie du COVID 19, ceux qui ont contracté la maladie et ont été guérir devait quand même porter un masque par la suite.

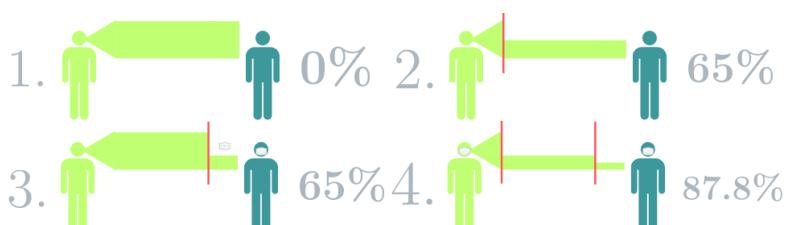
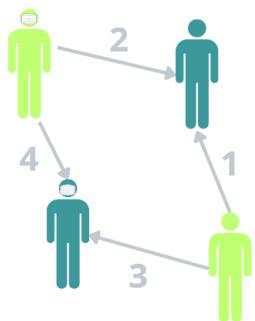


Figure 47- Efficacité des 4 cas de figures.

Figure 46- Cas 4 d'interactions.

Je vais considérer que 50% la population d'agents dans une région d'épidémie porte un masque lors de l'état d'urgence. Alors en moyenne en supposant que les agents interagissent de manière aléatoire les uns aux autres, ce qui est le cas, chaque cas précédent aura une probabilité de  $\frac{1}{4}$  d'être réalisé.

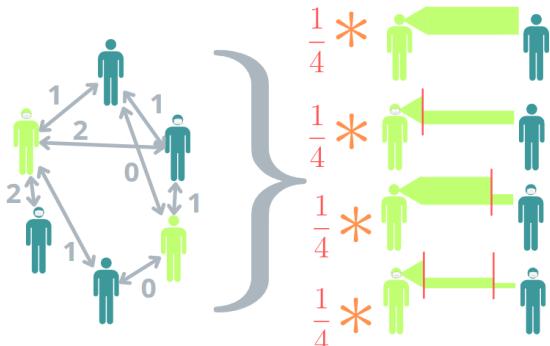


Figure 48- Probabilité des 4 cas de figures.

Là où on aurait pu penser intuitivement à une réduction globale de 32.5% ( $0.65 * 0.5$ ). Les masques sont beaucoup plus efficaces que ce que l'on pense.

On remarque que  $\frac{3}{4}$  des interactions un masque est présent. Ainsi en faisant la moyenne de ces quatre possibilités nous obtenons une réduction de transmission globale de 55%

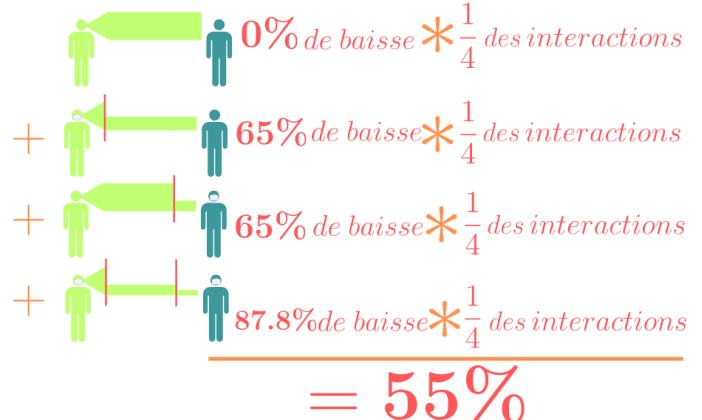


Figure 49- Efficacité général.

### 3.3.1 Port du masque sur toute la population

Le première ABMS avec un système de port de masque peut impacter l'ensemble des agents de la population, sans distinction régionale concernant l'épidémie. Par conséquent, le modèle ne régulera pas la mobilité des agents, contrairement au précédent ABMS.

Ainsi, le modèle imposera le port du masque de manière aléatoire à 50% de la population d'agents lors d'une situation d'urgence. Naturellement, une fois que la variation du nombre d'agents infectés aura atteint le seuil permettant de sortir de l'état d'urgence et que la période d'état d'urgence sera écoulée, le modèle retirera les masques à tous les agents concernés.

Au sein de la classe *Host*, des ajustements sont nécessaires pour les agents par rapport à la méthode d'infection (Fig.10). Comme mentionné précédemment, il existe 4 scénarios possibles, ce qui signifie que les probabilités de transmission par un agent infecté et de contracter la maladie par un agent susceptible peuvent varier en fonction du port du masque. Ainsi, pour chaque agent susceptible se trouvant dans la zone d'infection d'un agent infecté, si cet espace contient un agent infecté portant un masque, la probabilité d'infection est multipliée par 0,35 à chaque fois pour atteindre une efficacité de 65%. De même, si l'agent susceptible porte un masque, la probabilité de contracter la maladie est également multipliée par 0,35.

```
for agent in neighbors_of_susceptible_agent:
    if agent.is_infected:
        d = susceptible_agent.distance_to(agent)
        infection_probability = self.model.p_inf
        if agent.wearing_mask:
            infection_probability *= 0.35 # 65% effectiveness of the mask
        p *= (1.0 - infection_probability * self.phi(d))

contraction_probability = self.model.q_inf
if susceptible_agent.wearing_mask:
    contraction_probability *= 0.35 # 65% effectiveness of the mask
p = contraction_probability * (1.0 - p)
```

Figure 50- Modification d'une partie de la méthode infect.

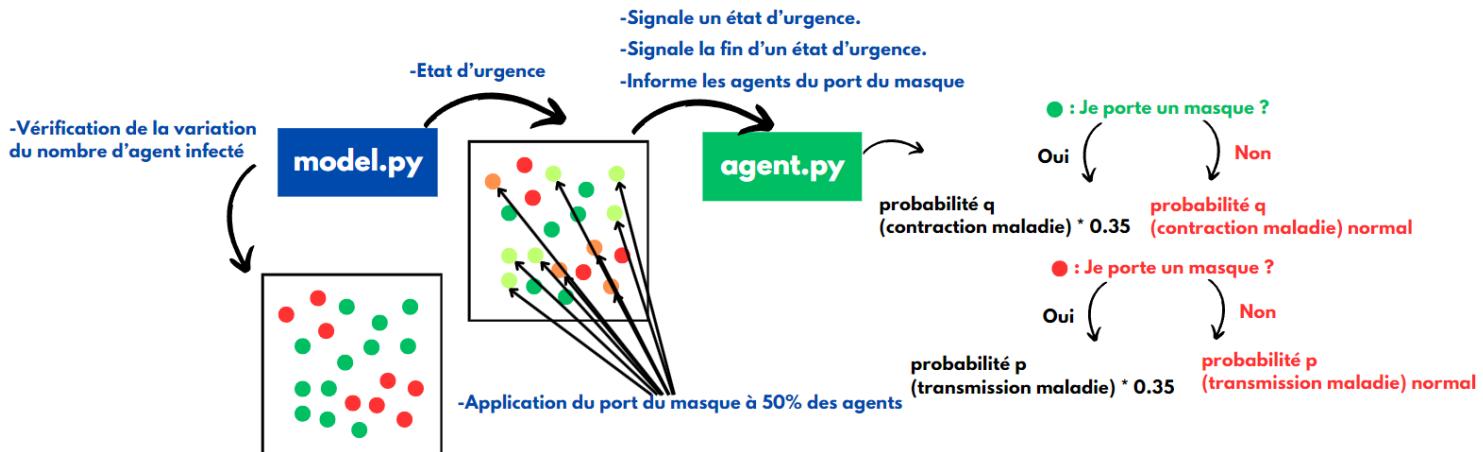


Figure 51-Fonctionnement système port du masque.

### 3.3.1.1 Résultat

Current Step: 176

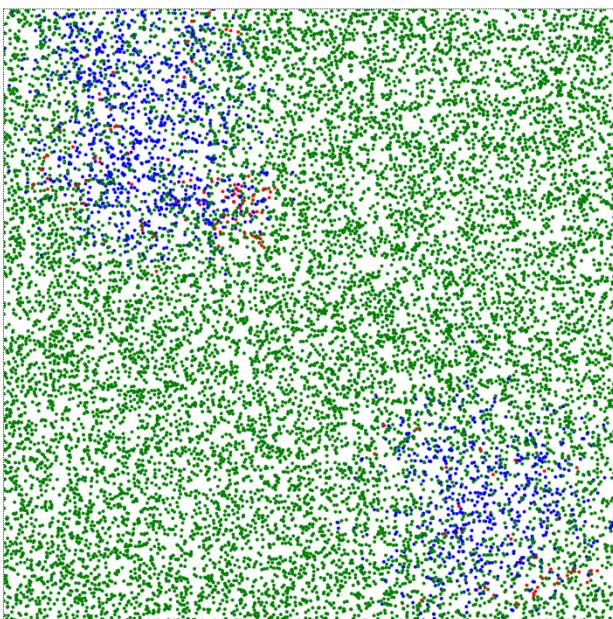


Figure 52.a – Population après 170 cycles.

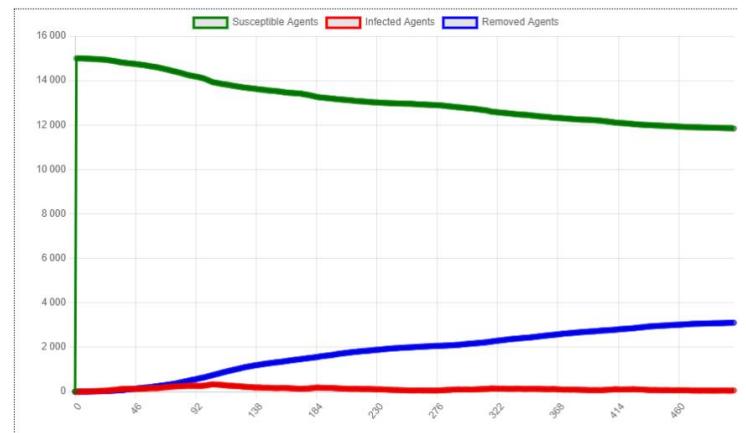


Figure 52.b – Dynamique de la population après la fin de la simulation au cycle 498.

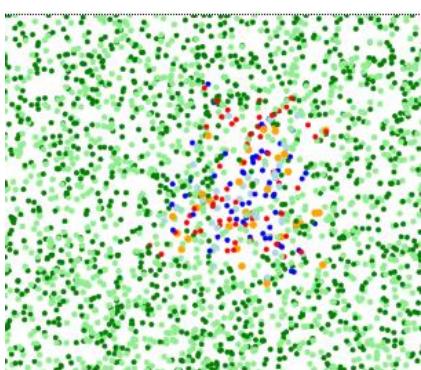


Figure 52.c – Dynamique de taux de variation d'infection.

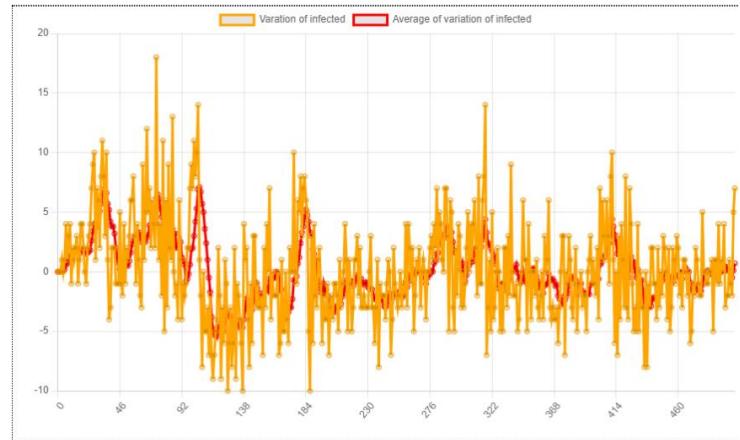


Figure 52.d – Exemple représentation port du masque. (Agent infecté avec masque : orange. Agent susceptible avec masque : vert clair. Agent « retiré » avec masque : bleu clair)

Il est observé que la propagation de l'épidémie diminue significativement, tout comme avec les ABMS précédents en période d'urgence, même si seulement 50% de la population porte un masque. Il est cependant à noter que cette diminution n'est pas aussi efficace qu'avec la réduction de mobilité sur les agents, mais j'ai pris l'hypothèse où 50% de la population porte un masque. De plus, il convient de noter qu'à la différence des ABMS impliquant une réduction importante de mobilité, les individus peuvent se déplacer librement moyennant le port du masque. Cependant, se pose la question de savoir s'il est nécessaire que tous les individus, même ceux éloignés des foyers d'épidémie, portent un masque. En effet, le port du masque peut être contraignant pour un individu, ce qui explique pourquoi certaines personnes ont refusé de le porter pendant la pandémie de COVID-19.

### 3.3.2 Port du masque dans les zones d'épidémies

Cette ABMS va reprendre le système de création de zones dynamique avec la méthode `update_epidemic_zone`, figure 41. Ainsi, le modèle imposera le port du masque à 50% de la population résidant dans chaque zone épидémique.

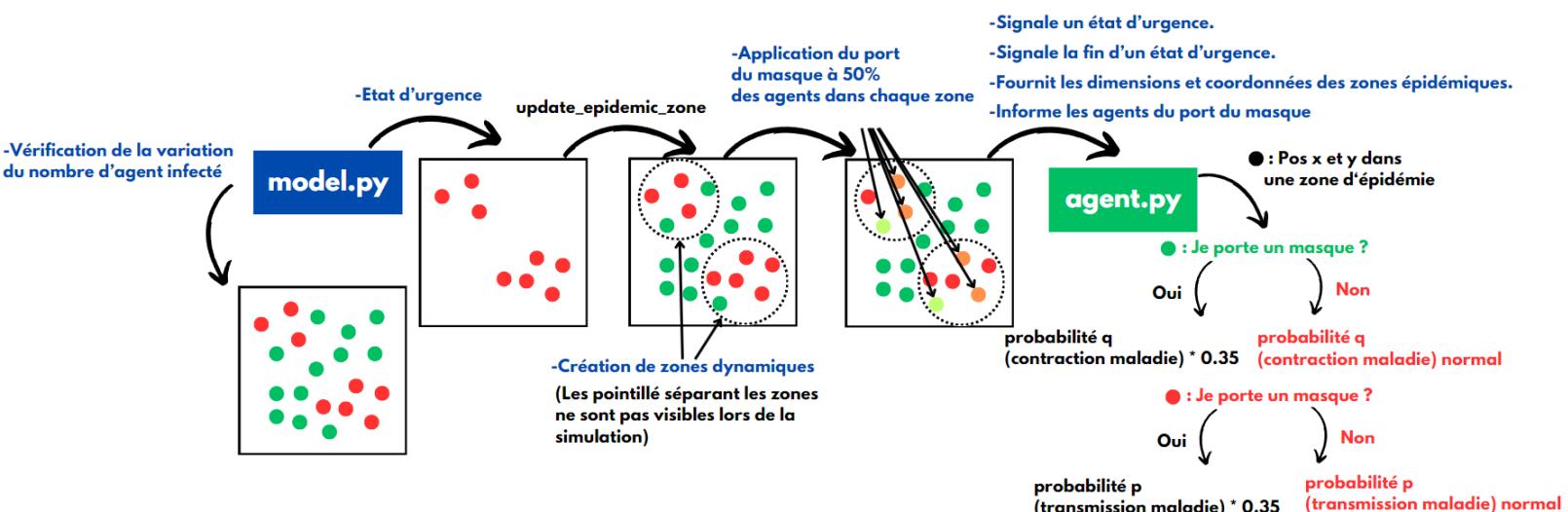


Figure 53 – Fonctionnement du système port du masque dans les zone d'épidémie dynamique.

#### 3.3.2.1 Résultat

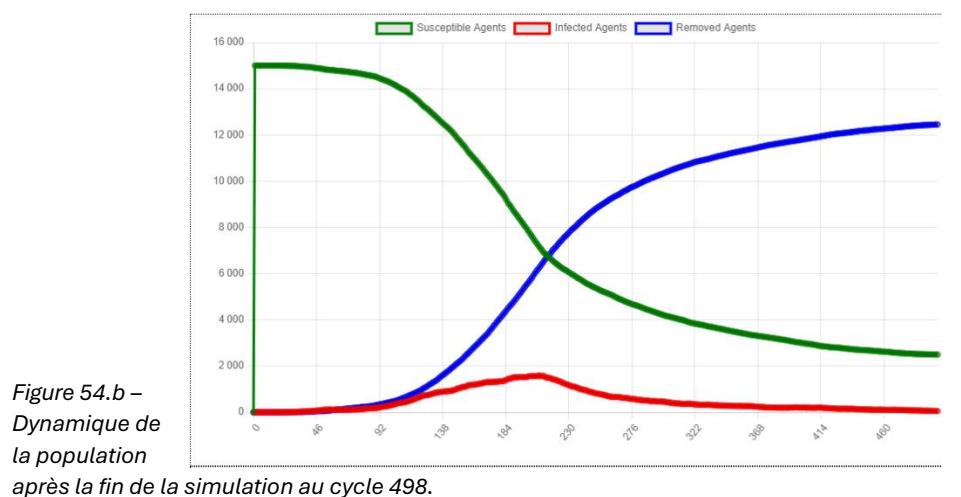
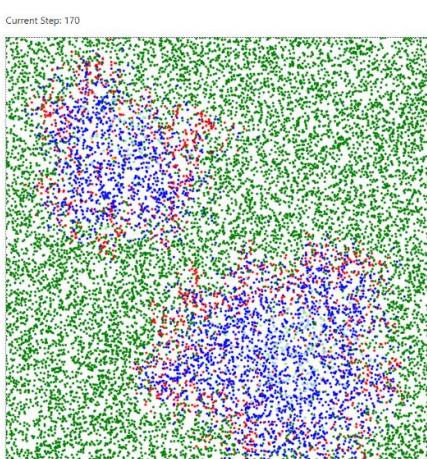


Figure 54.a – Population après 170 cycles.

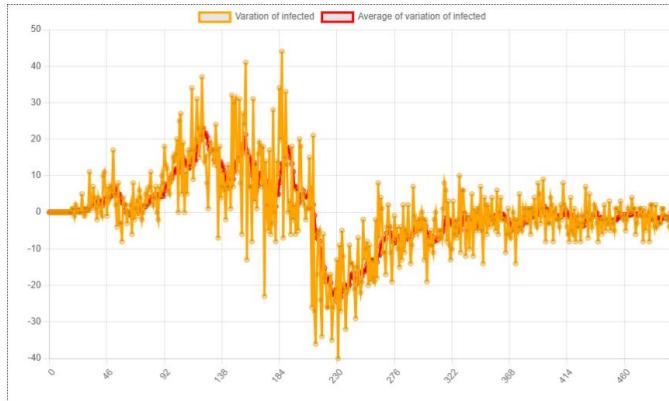


Figure 54.c – Dynamique de taux de variation d'infection.

La propagation ne diminue pas de manière suffisante avec cette ABMS. En effet, le modèle actuel prévoit que la moitié des agents présents dans une zone d'épidémie doivent porter un masque en cas d'urgence, tout en leur permettant de se déplacer librement. Cependant, si des agents infectés ne portent pas de masques et quitte une zone d'épidémie, ils risquent davantage de contaminer d'autres individus vulnérables créant ainsi d'autres zones épidémiques. Cette situation est problématique car le nombre d'agents infectés risque d'augmenter malgré l'état d'urgence en vigueur. De plus, étant donné que ce nombre ne diminue pas, le modèle n'est pas en mesure de mettre fin à l'état critique, posant ainsi un sérieux problème. Il serait donc indispensable que le port du masque soit étendu à l'ensemble de la population.

### 3.3.3 Port du masque dans des zones d'épidémies et mobilité réduite

Le récent ABMS instauré réintroduit la notion dynamique des zones épidémiques, incluant une mesure aléatoire concernant le port du masque par 50% des individus présents dans ces zones. Néanmoins, les agents évoluant au sein de ces zones ne verront pas leur vitesse affectée, mais seront contraints de demeurer dans la région d'épidémie jusqu'à la levée de l'état d'urgence. Ainsi, il suffit de revoir l'ABMS, (fonctionnement figure 53), antérieur et d'ajuster la logique des déplacements des agent

-Signale un état d'urgence.

-Signale la fin d'un état d'urgence.

-Fournit les dimensions et coordonnées des zones épidémiques.

-Informe les agents du port du masque

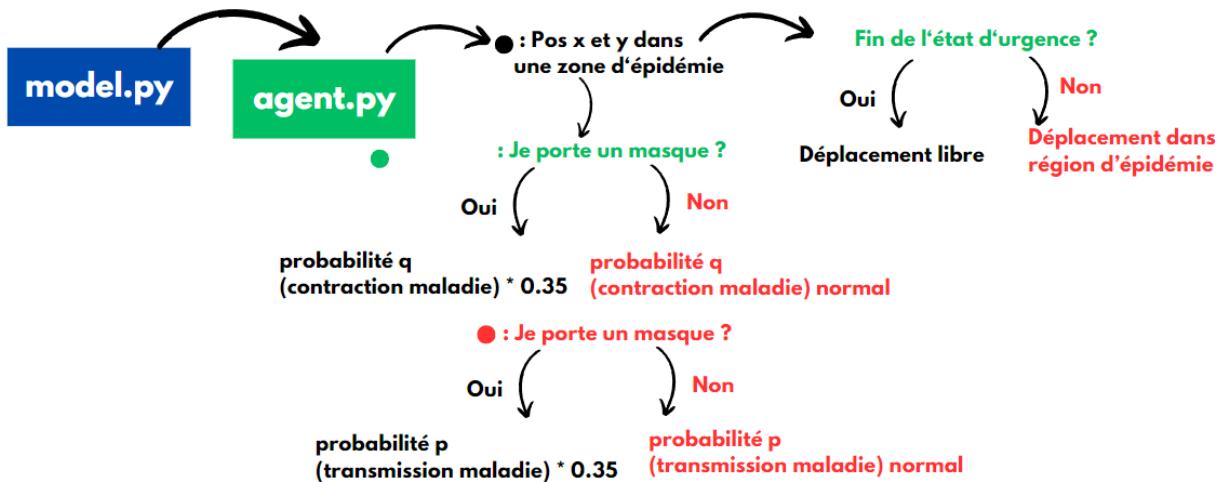


Figure 55 – Fonctionnement du système port de masque dans région d'épidémie avec limitation déplacement.

### 3.3.3.1 Résultat

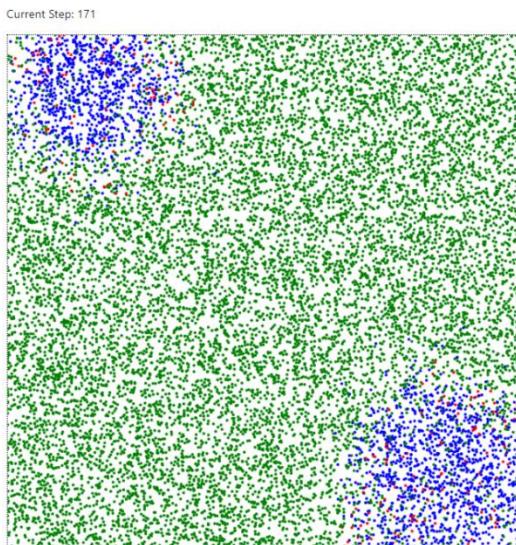


Figure 56.a – Population après 170 cycles.

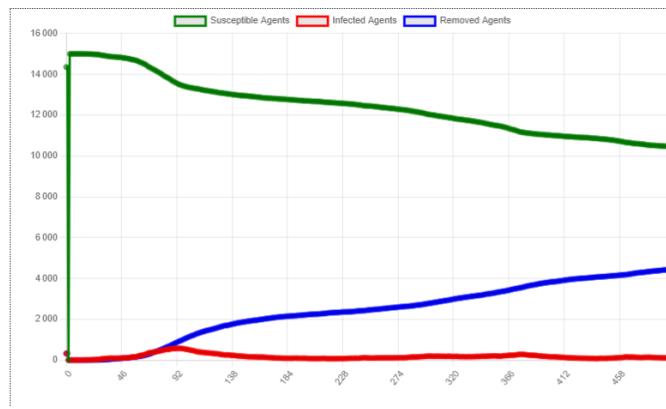


Figure 56.b – Dynamique de la population après la fin de la simulation au cycle 498.

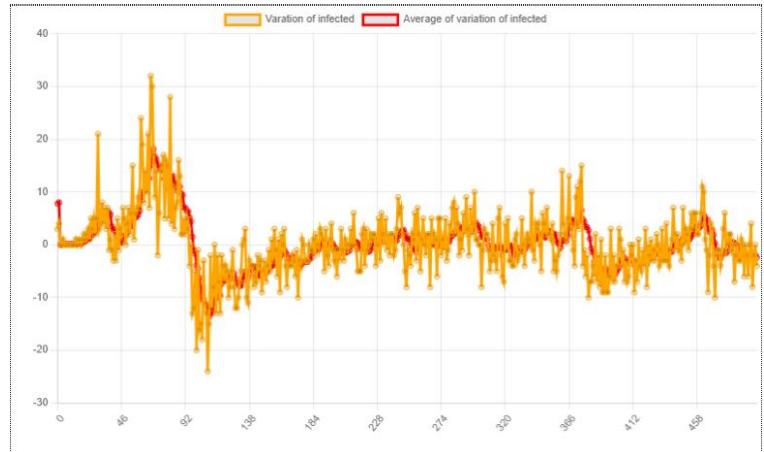


Figure 56.c – Dynamique de taux de variation d'infection.

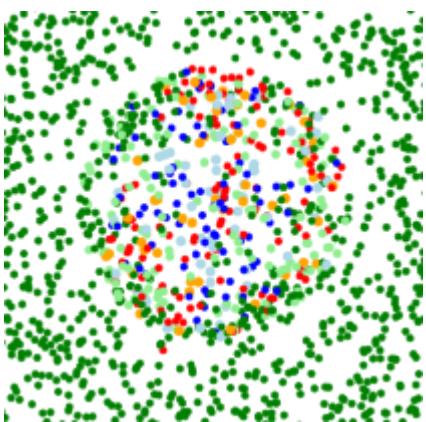


Figure 56.d – Exemple représentation de régions d'épidémie avec port du masque. (Agent infecté avec masque : orange. Agent susceptible avec masque : vert clair. Agent « retiré » avec masque : bleu clair)

Il est à noter que les résultats obtenus sont très similaires à ceux du premier système de masque. Un avantage notable par rapport à ce dernier réside dans le fait que seuls les agents présents dans des zones épidémiques portent un masque. Néanmoins, ces zones épidémiques restreignent la liberté de mouvement des agents qui s'y trouvent. Ce dispositif contribue significativement à contenir la propagation de l'épidémie en libérant un maximum d'agents au sein de la population et en ne mettant en place des mesures qu'au sein des foyers épidémiques.

## 3.4 Bilan

Concernant les ABMS avec des systèmes qui réduisent la vitesse de déplacement des agents pour limiter leur mouvement, il est à noter que cette mesure nécessite pas d'être appliquée à l'intégralité des individus d'une population, mais plutôt de cibler les foyers épidémiques. En ce qui concerne les ABMS avec des systèmes impliquant le port du masque, il apparaît que si celui-ci est généralisé à l'ensemble de la population, la propagation de l'épidémie diminue même si seulement 50% des individus le portent. Cependant, cette diminution peut également être observée en

concentrant les efforts sur les foyers épidémiques et en imposant le port du masque à 50% des résidents de chaque foyer tout en restreignant leurs déplacements hors de ces régions.

Une interrogation subsiste quant aux préférences des individus entre une réduction considérable des déplacements dans les zones touchées par l'épidémie ou bien le port du masque dans ces mêmes zones avec la liberté de mouvement préservée mais avec l'interdiction de sortir de ces zones.

Il est ainsi possible d'appliquer diverses mesures pour endiguer la propagation d'une épidémie. Il ressort que les systèmes de réduction de vitesse se révèlent plus efficaces que le port généralisé du masque. Toutefois, il convient de souligner que j'ai opté pour une application du masque à hauteur de 50% de la population et qui montre des résultats considérables.

## 4 Application Web

L'application web, conçue à l'aide de la bibliothèque Python Streamlit, permettra d'interagir avec les divers ABMS créés et de les démarrer directement depuis l'application, évitant ainsi la nécessité de saisir des lignes de commande dans un terminal.

### 4.1 Interface utilisateur

L'interface utilisateur se compose initialement des onglets correspondant à chaque page de l'application web, une pour chaque ABMS. Ainsi, pour lancer la simulation d'un ABMS spécifique, il convient de se rendre sur sa page dédiée.

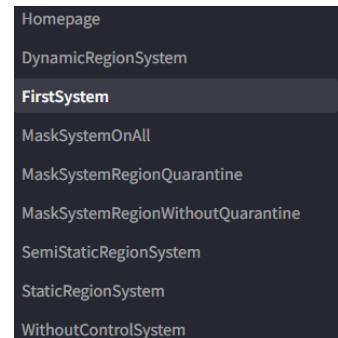


Figure 57 –Onglets de l'application web.

Un fois sur l'une des pages de l'application, l'utilisateur peut renseigner les valeurs du modèle de l'ABMS en question. Tous les ABMS auront au minimum les paramètres suivants modifiables : figure 16. En effet, dans les ABMS avec un systèmes de région dynamique il est possible de sélectionner un coefficient pour rayon des régions. Pour les ABMS doté d'un système de port de masque il est possible de choisir l'efficacité de celui-ci est à quel proportion de la population il s'applique.

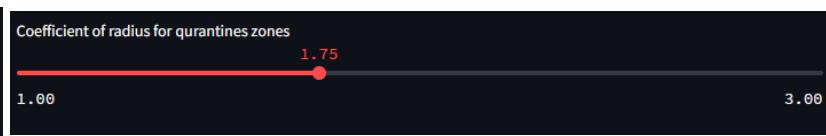


Figure 59 –Saisie coefficient du rayon des régions dynamique d'épidémie.

Figure 58 –Exemple de saisie des paramètres.

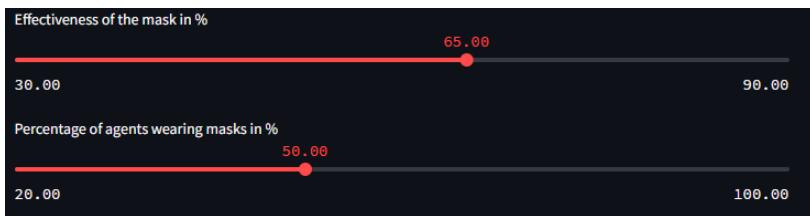


Figure 60 –Saisie efficacité + application du masque.

Ensuite, l'utilisateur doit indiquer la position des agents infectés en les plaçant directement dans un espace dont il a choisi la taille et hauteur pour le modèle d'ABMS concerné.

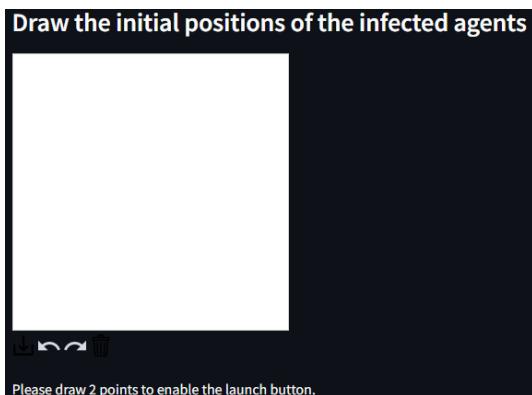


Figure 61 –Agents infecté pas encore placé sur l'environnement.

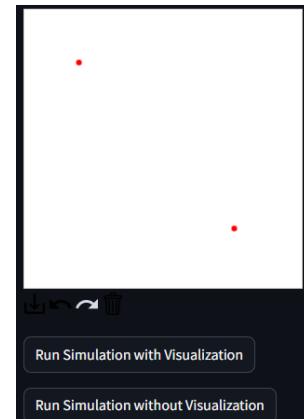


Figure 62 –Agents placés, boutons disponibles

L'utilisateur peut alors démarrer la simulation avec une visualisation en temps réel de la propagation de l'épidémie avec le déplacement des agents et des graphiques dynamiques grâce au premier bouton figurant sur la figure 46. Cela exécutera le fichier server.py correspondant à l'ABMS et ouvrira dans le navigateur de l'utilisateur une nouvelle page web sur un port libre grâce à MESA.

Il lui est également possible d'exécuter la simulation sans visualisation, c'est-à-dire sans aperçu visuel du déroulement. Contrairement à la simulation avec visualisation où il est possible d'interrompre et reprendre le processus à tout moment souhaité, celle sans visualisation doit être configurée avec un nombre prédéfini d'itérations.

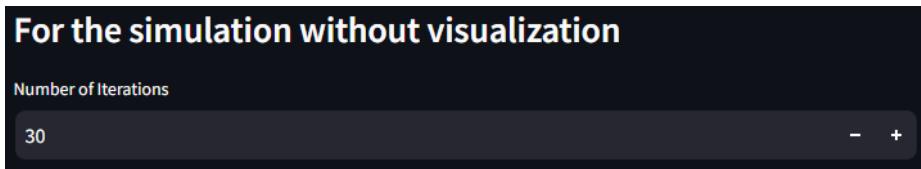


Figure 63 –Saisie du nombre d'itération.

Il est avantageux de lancer la simulation de cette manière en raison de la rapidité accrue de l'exécution. De plus, une fois la simulation sans visualisation terminée, des données et des graphes seront affiché sur la page de l'application web de l'ABMS concerné, demeurant visibles même en naviguant vers d'autres pages de ladite application. Cela permettra ainsi de comparer les résultats obtenus entre les différents ABMS. Néanmoins, du fait qu'il n'y a pas de visualisation,

l'environnement avec les agents ne sera pas affiché et il ne sera pas possible de démarrer/arrêter/réinitialiser ou modifier la vitesse de la simulation.

## 4.2 Eléments de comparaisons

Ces éléments comparatifs disponibles lors du lancement de la simulation sans visualisation permettront d'évaluer l'efficacité des différentes mesures de protections contre la propagation d'épidémie, entre eux. En effet, pour les ABMS dotés d'un système de contrôle sur la vitesse des agents, il sera possible de comparer le nombre de déplacement effectué par l'ensemble des agents pendant la simulation. Ainsi, pour chaque agent avec une vitesse normale, *wandering\_speed\_high*, le nombre de déplacement total augmentera de 1. En revanche, si un agent est soumis à une réduction du vitesse, *wandering\_speed\_low*, le nombre total de déplacement augmentera selon le calcul suivant :  $\frac{wandering\_speed\_low}{wandering\_speed\_high}$ .

Susceptible Agents	Infected Agents	Removed Agents	Total Steps
14598	75	329	1440191
↑ 15000	↑ 2	↑ 0	↑ 0

Figure 64 – Eléments de comparaison système réduction mobilité.

Pour les ABMS impliquant le port du masques, l'élément comparatif déterminant sera le nombre de masque total utilisés lors de la simulation.

Susceptible Agents	Infected Agents	Removed Agents	Total masks used
13643	195	1164	15002
↑ 15000	↑ 2	↑ 0	↑ 0

Figure 65 – Eléments de comparaison système avec masques.

Tout comme dans le cas des simulations avec visualisation, des graphiques seront disponibles pour illustrer l'évolution du nombre d'agents dans chaque état (S, I ou R), montrer comment varie le nombre d'agents infectés et présenter graphiquement comment évolue le nombre d'agents « retirés ».

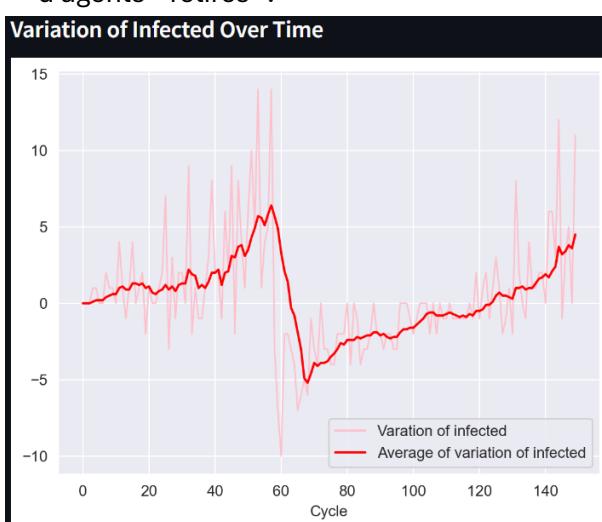


Figure 66 – Dynamique du taux de variation d'infection.

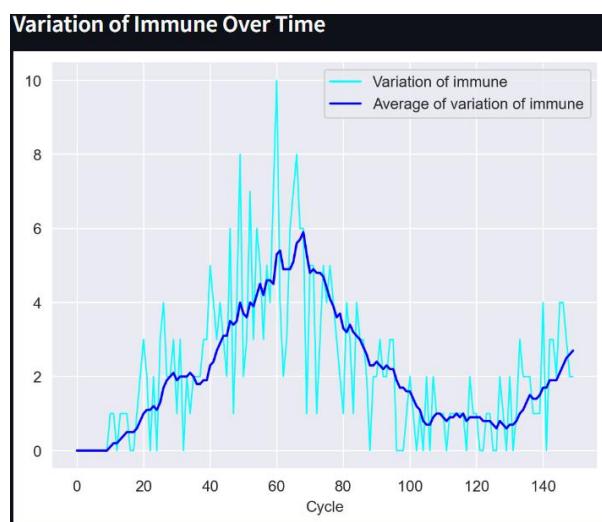


Figure 67 – Dynamique du taux de variation de guérison.

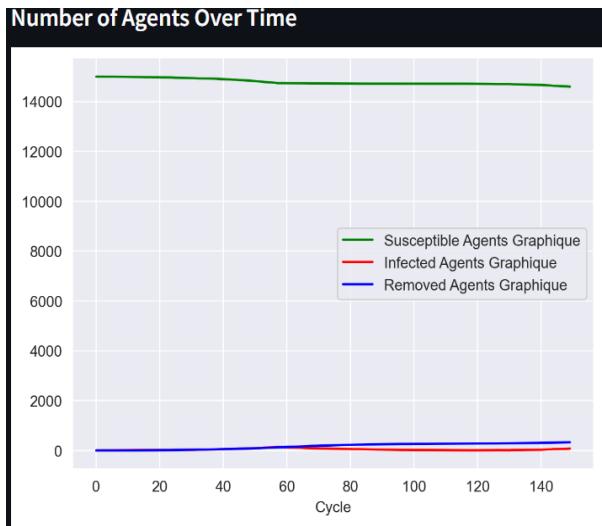


Figure 68 – Dynamique de la population.

### 4.3 Bilan

L'application web offre une interface intuitive et interactive pour gérer et exécuter les divers modèles d'ABMS. En centralisant ces simulations dans une application web, elle élimine la complexité d'utilisation des lignes de commande, rendant l'outil accessible même pour les utilisateurs non techniques. Cette application permet de lancer des simulations directement depuis l'interface avec des options de visualisation en temps réel, facilitant la compréhension de la dynamique des modèles. Les utilisateurs peuvent ajuster les paramètres facilement et exécuter les simulations avec ou sans visualisation, offrant ainsi une flexibilité et une optimisation des performances selon les besoins. De plus, les outils de comparaison et d'analyse des résultats permettent une évaluation efficace des mesures de protection contre la propagation des épidémies, renforçant l'utilité de l'application pour les chercheurs et les professionnels.

## Conclusion

Ce stage, d'une durée de 10 semaines, m'a permis de m'initier au domaine de la recherche. J'ai ainsi eu l'opportunité d'explorer le fonctionnement d'un modèle multi-agent spatial SIR dans le but de simuler la propagation d'une épidémie. J'ai été en mesure de comprendre, réutiliser et proposer une extension au travail de recherche de M. Bădică en utilisant un Framework approprié, MESA, tout en suggérant diverses stratégies de mesures préventives pour contenir la propagation d'une épidémie. Cette expérience m'a conduit à conclure qu'il n'est pas nécessaire de mobiliser chaque individu au sein d'une population pour stopper la propagation d'une épidémie, mais qu'il est plus judicieux de cibler les foyers épidémiques.

Par ailleurs, ce stage m'a offert l'occasion de concevoir ma première application web d'analyse pour un projet de recherche en utilisant le Framework Streamlit.

Toutefois, il convient de noter que les résultats des simulations pour chaque ABMS dépendent des paramètres choisis pour la simulation ; ainsi, toute modification de ces valeurs peut induire des résultats divergents. De plus, le modèle SIR formalise de manière générale la diffusion d'une maladie infectieuse en divisant la population en trois compartiments, ce qui constitue une simplification. Une approche plus élaborée aurait pu définir des paramètres spécifiques à une maladie donnée, et intégrer des variables telles que le genre, la taille et l'âge des individus pour obtenir une représentation plus fidèle à la réalité.

Ce stage s'est avéré extrêmement bénéfique à la fois sur le plan technique, en me permettant d'explorer le domaine de la recherche et d'acquérir de nouvelles compétences, et sur le plan personnel, en renforçant mon autonomie et mon adaptabilité. Il m'a enseigné à avoir un regard critique sur moi-même et à viser toujours plus haut. En suivant les directives de mon tuteur, j'ai pu développer ma capacité à prendre des initiatives tout en respectant les exigences du projet.

## Glossaire

**ABM (Agent-Based Model)** : Un modèle basé sur des agents est un modèle informatique permettant de simuler les actions et les interactions d'agents autonomes afin de comprendre le comportement d'un système et ce qui détermine ses résultats.

**Apache2** : serveur web open-source populaire.

**Application web** : logiciel applicatif hébergé sur un serveur et accessible via un navigateur web

**Cluster** : groupe de points de données similaires qui sont regroupés ensemble en fonction de leurs caractéristiques ou de leurs attributs dans le cadre d'une technique de clustering.

**Clustering** : technique d'apprentissage non supervisé qui regroupe des données similaires ensemble dans des clusters ou des groupes, en se basant sur leurs caractéristiques ou leurs attributs.

**DBSCAN (Density-Based Spatial Clustering of Applications with Noise)** : algorithme de clustering qui regroupe les points proches en fonction de leur densité, tout en identifiant les points anormaux comme du bruit.

**DOM (Document Object Model)** : représentation des éléments d'une page web sous une hiérarchie d'objets, où chaque balise est un nœud et dont le tout forme un arbre d'éléments. Ces éléments peuvent être modifiés avec du JavaScript.

**Donnée étiquetée** : donnée associée à une catégorie ou à une classe.

**Epidémie** : Développement et propagation rapide d'une maladie contagieuse, le plus souvent d'origine infectieuse, dans une population.

**Framework** : structure logicielle, canevas ou un ensemble d'outils et de composants logiciels à la base d'un logiciel ou d'une application, associés à un langage spécifique qui permettent de simplifier et d'uniformiser le travail des développeurs.

**Fichier** : ensemble de données stockées sur un support de mémoire informatique, identifiable par un nom unique et une extension.

**GAMA** : plateforme de modélisation et de simulation basée sur des agents (ABMS) qui permet aux utilisateurs de créer, exécuter et analyser des modèles complexes, où des agents autonomes interagissent dans un environnement pour simuler et étudier le comportement d'un système.

**Git** : système de contrôle de version décentralisé utilisé pour suivre les modifications apportées au code source et faciliter la collaboration entre développeurs.

**GitBash** : Git Bash est une application pour les systèmes Windows qui fournit une émulation de la ligne de commande Git, ainsi que la capacité d'exécuter des commandes Bash. Elle permet aux utilisateurs de travailler avec Git de manière similaire à ce qu'ils feraient sur un système Unix, facilitant ainsi la gestion des versions et le développement de logiciels.

**GitHub** : GitHub est un site web où les développeurs peuvent stocker et partager leur code. Il permet de collaborer facilement sur des projets en suivant les modifications et en travaillant ensemble.

**IDE (Integrated Development Environment)** : logiciel qui fournit aux développeurs une suite d'outils complets pour écrire, tester et déboguer du code.

**Intelligence artificielle** : domaine de l'informatique visant à créer des systèmes capables de réaliser des tâches qui nécessitent normalement l'intelligence humaine, telles que la reconnaissance de la parole, la prise de décision, la résolution de problèmes et la compréhension du langage naturel.

**Interface utilisateur** : dispositif matériel ou logiciel qui permet à un utilisateur d'interagir avec un produit informatique.

**Librairie** : ensemble de fonctions, de classes pré-écrites et réutilisables qui peuvent être utilisées par différents programmes pour effectuer des tâches spécifiques sans avoir à écrire le code à partir de zéro. Les librairies facilitent le développement en fournissant des outils déjà testés et optimisés.

**Machine learning (apprentissage automatique)** : branche de l'intelligence artificielle qui permet aux ordinateurs d'apprendre et de faire des prédictions ou des décisions basées sur des données, sans être explicitement programmés pour chaque tâche.

**Macroscopique** : Ce qui peut être observé à l'œil nu.

**MESA** : Framework de modélisation et simulation basée sur les agents, en python.

**Méso-scopique** : échelle intermédiaire entre le macroscopique de notre mode quotidien et le microscopique des atomes et des molécules.

**Microscopique** : Qui se fait au moyen du microscope.

**Modèle** : représentation informatique simplifiée d'un système ou d'un phénomène, utilisée pour étudier son comportement et ses interactions.

**Page web** : document électronique consultable sur Internet via un navigateur.

**Pandémie** : épidémie qui atteint un grand nombre de personnes, dans une zone géographique très étendue.

**Prophylactiques** : Qui prévient la maladie.

**Serveur web** : logiciel qui distribue des pages web aux utilisateurs sur Internet.

**Stochastique** : phénomène qui ne se prête qu'à une analyse statistique.

**Système discret** : système dont les variables évoluent par sauts discrets plutôt que de manière continue.

**Système multi-agent (SMA)** : ensemble de plusieurs agents autonomes qui interagissent entre eux pour résoudre des problèmes ou simuler des comportements complexes.

**Technique d'apprentissage non supervisée** : méthode d'apprentissage automatique où les algorithmes traitent des données non étiquetées pour trouver des motifs ou des structures cachées.

**Technique d'apprentissage supervisée :** méthode d'apprentissage automatique où les algorithmes sont entraînés sur des données étiquetées pour prédire des résultats futurs.

**Trello :** application de gestion de projet en ligne qui utilise des tableaux, des listes et des cartes pour organiser et collaborer sur des tâches.

# Bibliographie

## Références

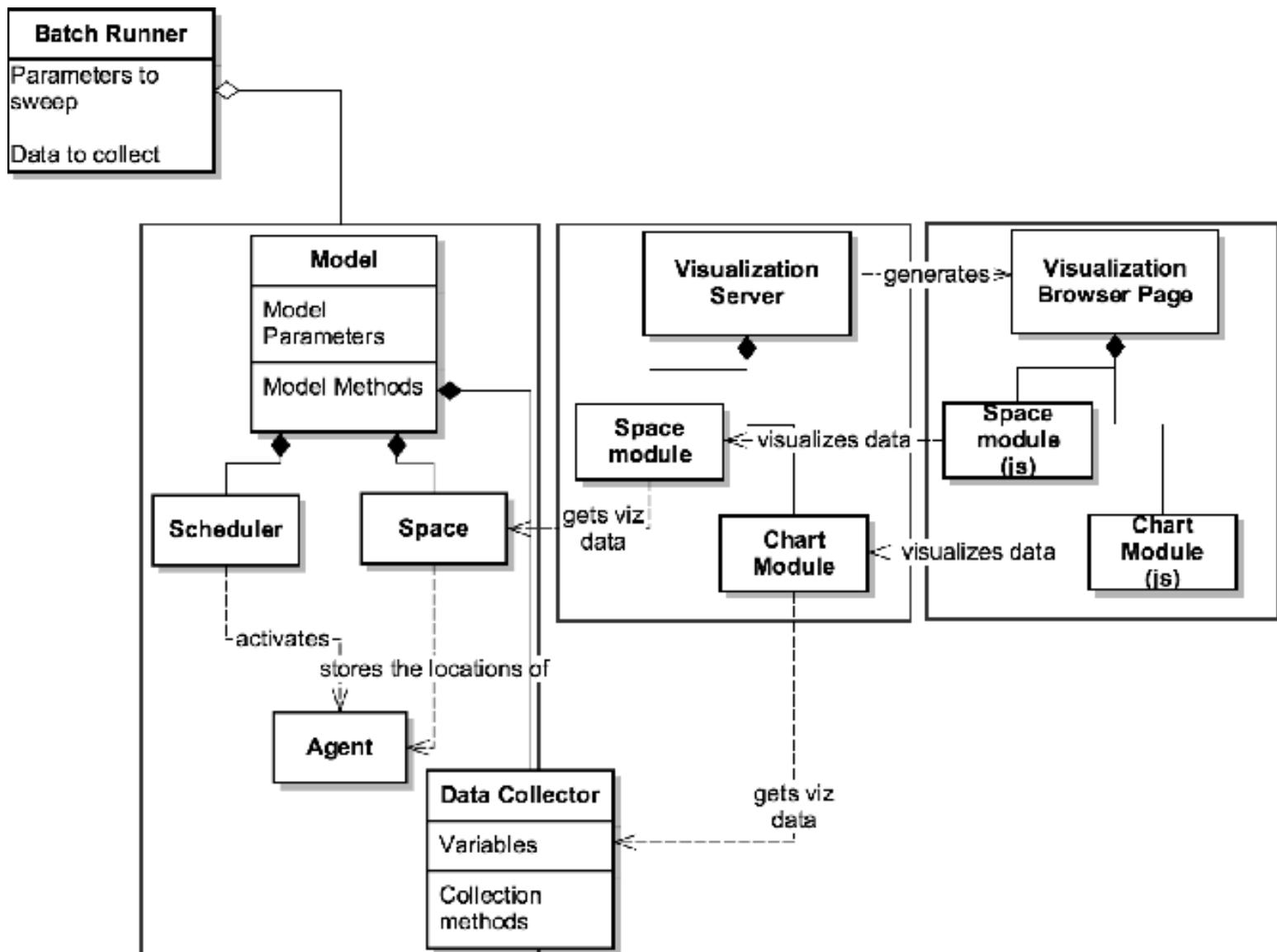
- [1] Article sur la Modélisation et Simulation Spatiale Basée sur le Modèle SIR Multi-Agents de la Gestion de la Propagation des Infections : [https://www.researchgate.net/publication/349658719\\_Multi-Agent\\_Simulation\\_of\\_Core\\_Spatial\\_SIR\\_Models\\_for\\_Epidemics\\_Spread\\_in\\_a\\_Population](https://www.researchgate.net/publication/349658719_Multi-Agent_Simulation_of_Core_Spatial_SIR_Models_for_Epidemics_Spread_in_a_Population)
- [2] Université de Craiova : <https://www.ucv.ro/en/> , <https://www.universityguru.com/fr/universites--roumanie>
- [3] Streamlit : <https://streamlit.io>
- [4] MESA : <https://mesa.readthedocs.io/en/stable/> , <https://github.com/projectmesa/mesa-viz-tornado>
- [5] K-Means : <https://blent.ai/blog/a/k-means-comment-ca-marche> , <https://fr.wikipedia.org/wiki/K-moyennes> , <https://datascientest.com/algorithme-des-k-means> , <https://mrmint.fr/algorithme-k-means>
- [6] Algorithme de Lloyd : [https://fr.wikipedia.org/wiki/Algorithme\\_de\\_Lloyd-Max](https://fr.wikipedia.org/wiki/Algorithme_de_Lloyd-Max) , <https://datascientest.com/algorithme-des-k-means> ,
- [7] K-Means ++ : <https://en.wikipedia.org/wiki/K-means%2B%2B> , <https://www.geeksforgeeks.org/ml-k-means-algorithm/> , <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html> .
- [8] DBSCAN : [https://fr.wikipedia.org/wiki/DBSCAN#:~:text=DBSCAN%20\(density-based%20spatial%20clustering.Jörg%20Sander%20et%20Xiaowei%20Xu.https://openclassrooms.com/fr/courses/4379436-explorez-vos-donnees-avec-des-algorithmes-non-supervises/4379571-partitionnez-vos-donnees-avec-dbscan](https://fr.wikipedia.org/wiki/DBSCAN#:~:text=DBSCAN%20(density-based%20spatial%20clustering.Jörg%20Sander%20et%20Xiaowei%20Xu.https://openclassrooms.com/fr/courses/4379436-explorez-vos-donnees-avec-des-algorithmes-non-supervises/4379571-partitionnez-vos-donnees-avec-dbscan)
- [9] Score silhouette : <https://blent.ai/blog/a/k-means-comment-ca-marche> , [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette\\_score.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html) ,
- [10] Trello : <https://trello.com/b/aLbuyhoF/projet-stage>

## Sources figures

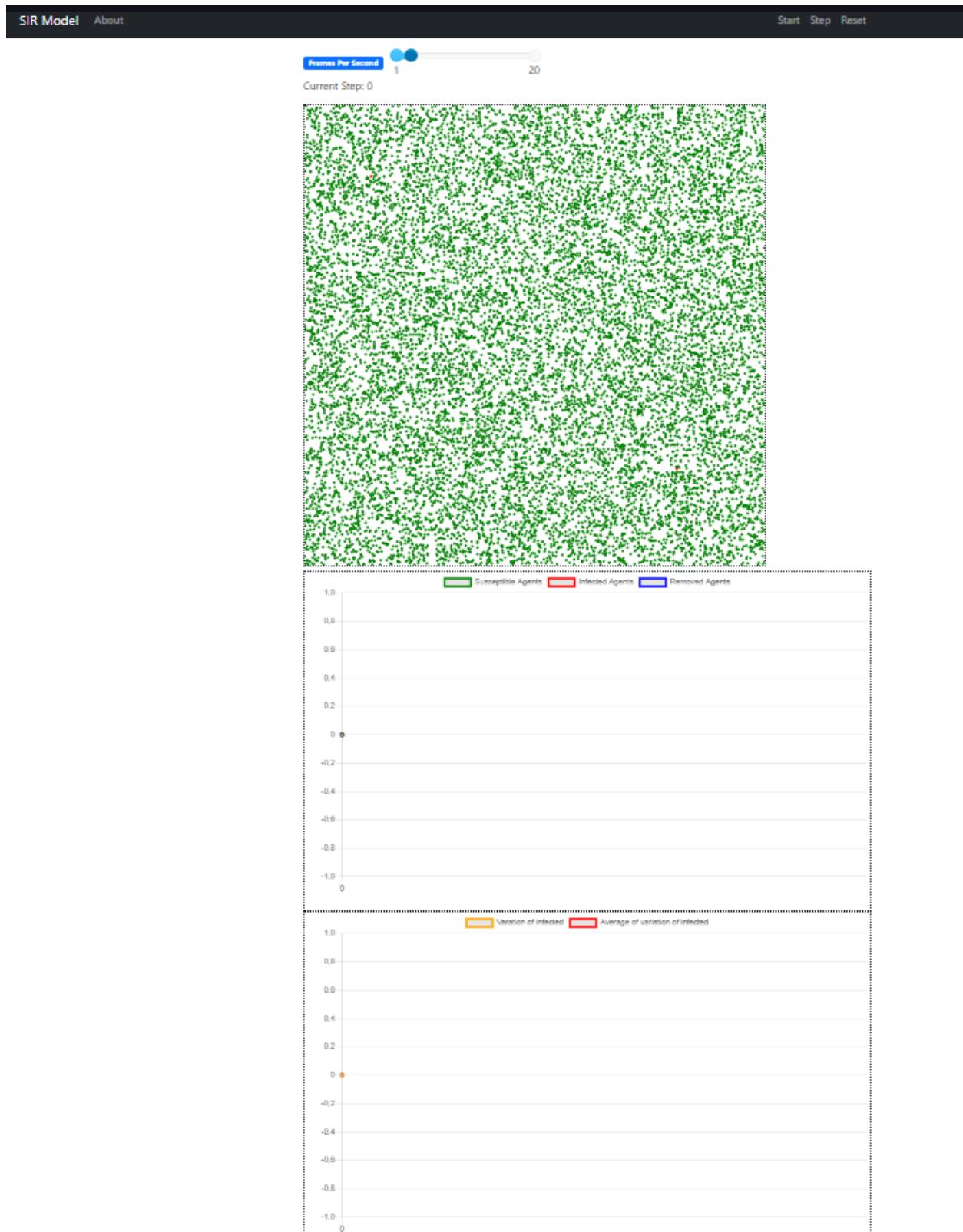
- [Figure 3]: <https://trello.com/b/aLbuyhoF/projet-stage>
- [Figures 5-6-7]: [https://www.researchgate.net/publication/349658719\\_Multi-Agent\\_Simulation\\_of\\_Core\\_Spatial\\_SIR\\_Models\\_for\\_Epidemics\\_Spread\\_in\\_a\\_Population](https://www.researchgate.net/publication/349658719_Multi-Agent_Simulation_of_Core_Spatial_SIR_Models_for_Epidemics_Spread_in_a_Population)
- [Figure 29] : <https://blent.ai/blog/a/k-means-comment-ca-marche>
- [Figure 30] : <https://youtu.be/fxOHbT9B4iA?si=ZMiDqBnaZlPpD96b>
- [Figures 31-32-33-34-35] : <https://youtu.be/4b5d3muPQmA?si=X7ZP5aFaRdVF1n5y>
- [Figures 36-37-38-39-40] : <https://youtu.be/HatwtJSsj5Q?si=-xvPXi-6F5kyU810>
- [Figure 44] : [https://www.hygiences.net/system/files/inline-images/HY\\_XXIX\\_2\\_Pittet\\_fig1.jpg](https://www.hygiences.net/system/files/inline-images/HY_XXIX_2_Pittet_fig1.jpg)
- [Figures 45-46-47-48-49] : <https://youtu.be/XMWidBXCxPY?si=cXqfjLkd19L0ts-U>

## Annexes

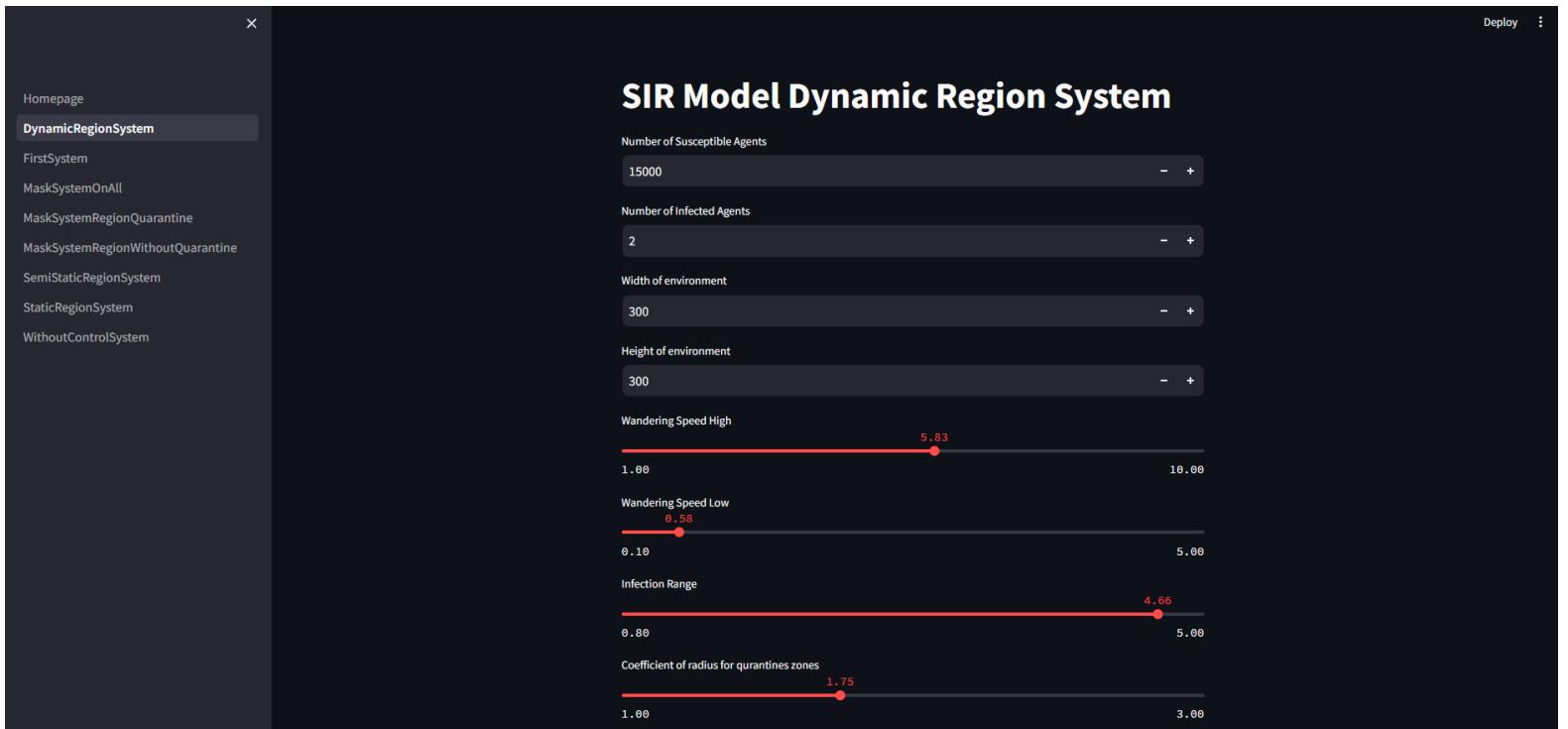
Annexe 1 – Structure Framework MESA.



## Annexe 2 – Page web ABMS MESA.



Annexe 3 – Application web Streamlit 1.



Annexe 4 – Application web Streamlit 2.

