Introduction aux Processus Stochastiques.
Projet 1: Analyse de la propagation d'un virus au sein d'un réseau

François Lievens : 20103816

2019-2020

1 Introduction

1.1 Introduction généale

Réalisé dans le cadre du cours d'introduction aux processus stochastiques, ce a pour but de modéliser la propagation d'une maladie virale au sein d'une population déterminée par le biais d'une chaîne de Markov en temps discret.

En effet, dans le contexte actuel de l'épidémie de COVID-19, il apparaît plus que jamais important de pouvoir modéliser l'évolution d'une épidémie. D'une part car nous devons nous apprêter à faire face à cette évolution, et d'autre part afin de simuler l'impact que peuvent avoir les diverses mesures que nous pouvons prendre contre la propagation de cet agent viral.

L'étude du modèle exact, réalisée sur deux populations de petite taille, sera développée dans la première partie de ce rapport. Nous allons ensuite approcher ce modèle par l'intermédiaire de simulations, permettant ainsi son étude sur des populations de tailles beaucoup plus grandes.

1.2 Notes préliminaires au sujet du code

Le programme implémenté afin de répondre aux diverses parties du projet a été rédigé en Java. Les différentes classes créées se trouvent dans le répertoire **src/** contenu dans l'archive du projet. Ce répertoire contient également une classe **Main.java** conçue de façon à exécuter les diverses méthodes générant les données exposées dans cette rédaction. Son exécution dans son entièreté met environ 120 secondes (temps réalisé sur les machines ms800). Les différentes actions exécutées sont détaillées à l'écran au cours de l'exécution. La compilation peut être exécutée par la commande suivante :

```
javac *.java
```

Et son exécution peut être réalisée, après compilation par la commande :

```
java Main
```

Ces deux commandes doivent être exécutées à partir du dossier **src/**. Ce dossier contient également le sous dossier **src/outputData/**. C'est ce dossier qui est utilisé pour exporter les données générées lors de l'exécution de la classe **Main**.

Le contenu des autres classes sera détaillé au cours de ce rapport.

2 Étude du modèle exact

Dans cette partie, nous allons nous intéresser au cas de deux populations de petite tailles qui sont représentées par les matrices d'adjacences W^{lin} pour laquelle $W^{lin}_{i,j} = 1$ si et seulement si $|i-j| = 1$, et W^{full} tel que $W^{full}_{i,j} = 1$ si et seulement si $i \neq j$.

Pour rappel, une matrice d'adjacence permet de représenter les connexion entre les noeuds d'un graphe. Chaque élément $W_{i,j}$ de la matrice représente une arête du graphe. Ces arêtes n'étant pas pondérées, une valeur de $W_{i,j}$ différente de zéro signifie donc, dans notre cas, que les individus i et j ont des contactes susceptibles de transmettre le virus entre eux.

2.1 Le modèle est-il un processus de Markov ?

Tout d'abord, il faut rappeler ce qu'est un processus stochastique : Il s'agit d'un ensemble de variables aléatoires, caractérisant généralement l'évolution de l'état système au cours du temps. Si ces variables sont indexées par l'ensemble des Naturels, nous pouvons dire que c'est un système stochastique à temps Discret.

Pour modéliser la propagation d'une maladie virale au sein d'une population par un processus stochastique, nous devons commencer par envisager la description des états de la population étudiée, et la façon dont ces derniers peuvent évoluer.

Ainsi, dans la situation ou nous voulons représenter l'évolution d'une maladie virale au sein d'une population, chaque individu de la population peut se trouver dans trois états individuels différents : I (infectieux), R (résistant) ou S (naïf).

L'état global de la population, étudiée à une unité de temps discret déterminée t, peut donc être modélisé par l'ensemble des états individuels de chacun des individus qui la composent.

En pratique, cela sera modélisé par un tableau de variables de type *char*, indexé de la même façon que la matrice d'adjacence de la population (et donc des individus qui la composent) et dont chaque élément peut prendre une des trois valeurs citées plus haut.

Si nous revenons à notre processus stochastique, nous pouvons donc déterminer son **espace d'états**, représentant l'ensemble des valeurs d'états qu'il peut prendre à chaque instant, comme étant l'intégralité des combinaisons possibles des états individuels des membres de la population étudiée.

Prenons par exemple une population de trois individus tous connectés entre eux. Étant donné que chaque individu a trois états individuels possibles (I, R et S), notre processus stochatique peut prendre 3^N états différents, et donc 27 états différents.

Afin de générer la composition de cet espace d'états, j'ai implémenté la méthode **statesFiller()** de la classe **ModeleExact.java** qui va générer toutes les combinaisons possibles à partir de la matrice d'adjacence du graphe représentant la population étudiée. Les voici pour notre population exemple de trois individus :

SSS	ISS	SIS	SSI	RSS	SRS	SSR	IIS	SII
ISI	RIS	IRS	SRI	RSI	ISR	SIR	RRS	SRR
RSR	III	RII	IRI	IIR	RRI	IRR	RIR	RRR

FIGURE 1

Lorsque notre système est dans un état X à un temps t_i, il est aisé de déterminer les états qu'il peut potentiellement prendre au temps suivant. En effet, chaque individu étant dans un état I a une probabilité μ d'être guéri en t_{i+1}, et donc de devenir R (et donc une probabilité $(1-\mu)$ de rester I) et a une probabilité β de contaminer respectivement chacun de ses voisins en t_{i+1}.

Ainsi, l'état que le système va prendre en t_{i+1} dépend uniquement de son état au temps t_i. Notre processus stochastique est donc un cas particulier que l'on appelle Chaîne de Markov, et dont nous allons pouvoir utiliser les propriétés pour l'étudier.

Dans une chaîne de Markov, nous pouvons déterminer les probabilités de tout potentiel état dans lequel le système peut se trouver en t_{i+1} par le calcul de la **matrice de transition**. En effet, l'élément $T_{i,j}$ de cette matrice, est la probabilité, si le système est dans l'état i au temps t_i de passer à l'état j au temps t_{i+1}.

Pour générer cette matrice de transition, la méthode **transitionFiller()** de la classe **modeleExact.java** a été implémentée. Cette dernière va utiliser l'espace d'états du système, préalablement généré par la méthode **statesFiller()**. Pour chaque état de cet espace, la méthode va générer un vecteur de la même taille que le vecteur d'état étudié, ne contenant que des zéros, ou des 1 aux index correspondants à des individus dont l'état individuel est I. En multipliant ce vecteur par la matrice d'adjacence de la population, nous obtenons donc un vecteur indexé de la même façon que la matrice d'adjacence de la population, et qui pour chaque individu relié directement à des individus infectés, va compter depuis combien d'individus infectés il peut être contaminé.

Ainsi, pour chaque noeud du graphe de la population dont la valeur dans ce vecteur est différente de zéro, si ce noeud est de statu individuel S, nous pouvons calculer sa probabilité d'être infecté en t_{i+1} par la formule $\beta * \sum_{i=1}^{N}(1-\beta)^{(i-1)}$, où N représente le nombre de voisins infecté auxquels cet individu est connecté. De la même façon, nous pouvons calculer les probabilités respectives qu'ont chaque individus de statu individuel I d'être guéri à l'unité de temps supérieure (et donc de devenir R) par la valeur de μ.

Maintenant que nous avons calculer les probabilités respectives des divers états individuels que vont pouvoir prendre les individus de la population au temps suivant, et vu qu'un état du système est une combinaison des états individuels des membres de cette population, nous pouvons calculer la probabilité de chacun des états que le système peut prendre à la prochaine unité de temps. En effet, la probabilité d'un état que le système peut prendre d'être un successeur de l'état actuel se calcul par le produit des probabilités de chacun des états individuels calculés ci-dessus. Nous pouvons donc compléter notre matrice de transition.

Une des vérifications que l'on peut réaliser sur les résultats obtenus est de multiplier la matrice de transition par un vecteur ne contenant que des 1. En effet, notre matrice doit être stochastique, c'est à dire que la somme des éléments de chaque lignes doit être égale à 1.

Une fois la matrice de transition générée, nous pouvons remarquer que toute réalisation du processus va inexorablement mener à un état permanent ne contenant plus aucun individu I. Par conséquent, notre chaîne n'est pas irréductible et absorbante, chaque état final étant un état absorbant (dont la probabilité de rester dans cet état à l'unité de temps suivante est de 1).

Il est à noter que, comme nous le verrons plus loin, il est possible d'obtenir la matrice de transition sous sa forme canonique afin d'observer de façon plus visuelle les types d'états. Pour obtenir la matrice sous cette forme, les états générés sont préalablement triés par la méthode **statesSorter()** avant d'utiliser la méthode **transitionFillr()**

Il est aisé de remarquer que lorsque l'on quitte un état qui n'est pas final, nous n'allons jamais retourner dans cet état. Cela est dû au fait qu'un individu dans un état I ne peut pas redevenir S.

2.2 Durée moyenne de la maladie pour un individu

La probabilité de guérison au temps suivant d'un individu infecté est donnée par la probabilité μ, la distribution du temps nécessaire à une guérison peut être représentée par une loi géométrique tel que la probabilité d'être guéri au temps k peut être déterminée par la formule : $P(X = k) = (1-\mu)^{k-1} * \mu$.

Ainsi, le temps moyen avant guérison correspond à l'espérance mathématique : $E[X] = \frac{1}{\mu}$. Si l'on considère le cas ou $\mu = 0.2$, il faudra donc en moyenne 5 unités de temps avant guérison d'un individu contaminé.

2.3 Évolution des courbes épidémiques

Pour chaque réalisation du processus, et à chaque unité de temps discret parcourue, nous pouvons qualifier l'état du système par trois valeurs issues du comptage du nombre d'individus de la population de statut individuel respectif I, R et S. Lorsque nous représentons ces données sous forme de graphique avec le temps en abscisse, nous obtenons les trois courbes que nous appellerons **courbes épidémiques**.

Les données nécessaires pour dessiner ces courbes sont générés par la classe **ModeleExact.java** dont le constructeur est appelé avec la matrice d'adjacence de la population à étudier en paramètre, ainsi que la valeur de β et de μ. Les matrices d'adjacences W^{full} et W^{lin}, décrites dans l'introduction du point 2, sont quant à elles générées par les méthodes **graphGeneratorLin()**

et **graphGeneratorFull()** prenant en paramètre la taille de la population à générer (6 dans notre cas).

L'objet de type **ModeleExact** permet ainsi d'étudier le modèle par les étapes suivantes :

1. La méthode **statesFiller()** génère tous les états possibles du système à partir de la matrice d'adjacence comme décrit précédemment.
2. La méthode **statesSorter()** va trier les états de façon à placer les permanents à la fin de la liste et les initiaux au début. En effet, nous pouvons considérer tout état ne contenant plus d'individu infectieux comme étant un état absorbant. Ainsi la matrice de transition obtenue à l'étape suivante sera directement sous forme canonique. La méthode **transitionFiller()** génère la matrice de transition. Cette dernière étant sous forme canonique, nous pouvons en extraire facilement certaines partitions afin d'en faciliter l'étude. Si P est la matrice de transition sous forme canonique, nous pouvons en extraire les partitions suivantes :

$$P = \begin{pmatrix} \mathrm{P}_T & \mathrm{P}_{T,P} \\ 0 & \mathrm{P}_P \end{pmatrix}$$

   Toutes ces matrices, si nous le souhaitons, sont exportable au format *.csv* par la méthode **exportToCSV()** appliquée aux objet de type **Matrix** et avec le nom du fichier en paramètre.
3. La matrice fondamentale (R_T) peut alors être calculée : $R_T = (I - P_T)^{-1}$ en utilisant les outils d'opérations matricielles implémentées ans la classe **Matrix.java**.
4. La matrice fondamentale nous permet ainsi de calculer le temps moyen avant disparition du virus, qui correspond en fait au nombre d'unités de temps discret nécessaires avant de tomber dans un état absorbant. Ce calcul est implémenté par la méthode **ModeleExact.averageTimeBeforeEnd()** qui va sommer les éléments des lignes de la matrice fondamentale correspondant aux états initiaux, et en calculer la moyenne. Cela est exacte du fait que chaque état initial est équiprobable dans notre modèle.

Une fois l'objet de type **ModeleExact** construit, nous pouvons en déterminer les courbes épidémiques suivantes en utilisant sa méthode **EpidemicCurvesComputer()** prenant en paramètres le nombre maximal d'unités de temps à traiter et exportant un fichier *.csv* contenant les données des courbes. Pour procéder, cette méthode va, pour chaque unité de temps discret successive, élever la matrice de transition à la puissance supérieure et réaliser un produit vectoriel avec le vecteur représentant l'état initial. Pour chaque élément non-nul du vecteur obtenu, la méthode va compter le nombre d'individu de chaque état individuel (I, R ou S), qu'elle va pondérer par la probabilité obtenue et ajouter dans les vecteurs de comptage. Les résultats obtenus sont présentés ci-dessous

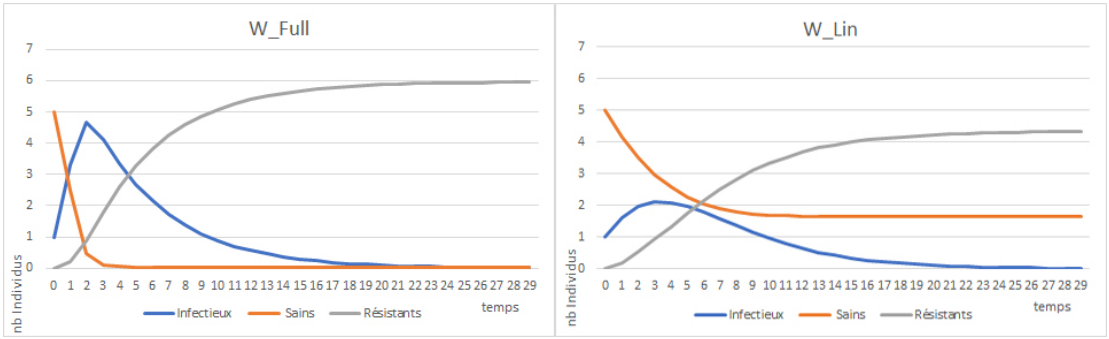

FIGURE 2

Les graphes de la figure 2 représentent en orange la proportion d'individus sains de la population, et donc d'individus qui ne sont ni infectieux, ni résistants. En gris la courbe des individus résistants, et qui donc ont été infecté par la maladie puis ont guérit, et en bleu les individus infectieux.

Nous pouvons dores et déjà remarquer que les courbes épidémiques sont fort différentes entre les deux populations. En effet, W^{full} représentant une population de six individus tous connectés, la maladie se transmet très rapidement à l'ensemble de la population, avec pour conséquence un pic épidémique très élevé et rapide, représentant une forte proportion de la population malade en même temps. Dans une population plus grande, nous pourrions envisager l'augmentation de cette courbe sous la forme d'une exponentielle. Par contre, si l'on regarde W^{lin}, où chaque individu est connecté au plus à deux autres, nous remarquons, comme nous pouvions nous y attendre, une propagation beaucoup plus lente. Cette propagation peut même s'essouffler avec comme conséquence, en fin d'épidémie, qu'une proportion beaucoup plus faible de la population soit passée par un stade infectieux.

2.4 Calcul du temps moyen nécessaire à la disparition du virus

Le calcul du temps nécessaire à la disparition de la maladie est réalisé à partir de la matrice fondamentale tel que décrit dans le point précédent. Il est calculé par la méthode **averageTimeBeforeEnd()**

Pour procéder, cette méthode va utiliser la matrice fondamentale R_t du processus, calculée à partir de la matrice de transition. Chaque ligne correspondant à un état initial du système, nous pouvons en sommer les éléments. En effet, lorsque l'on regarde l'élément $(R_t)_{i,j}$ de cette matrice, nous pouvons interpréter cette valeur comme étant le nombre moyen de passages par l'état j en partant de l'état i, et avant absorption par un état final.

Une fois les données générées, nous remarquons que l'épidémie a duré en moyenne plus longtemps pour W^{full} **(12,8183 unités de temps)** que pour W^{lin} **(11,6860 unités de temps)**. Cette différence est due à la forme "linéaire" de la population W^{lin} et à la disparition rapide de la maladie qu'elle engendre.

3 Étude sur base de simulations

L'étude du modèle exact demande d'énormes ressources en calcul informatiques. En effet, la création de sa matrice de transition a une complexité de l'ordre de $\Theta(n^3)$ pour une population de n individus. Cela était donc envisageable pour une population de six individus comme réalisé dans la première partie du rapport, mais impossible si l'on veut simuler la propagation de l'épidémie dans des populations plus grandes, comme c'est généralement le cas.

Une autre approche du problème consiste alors en un algorithme qui va générer aléatoirement un grand nombre de réalisation de la chaîne de Markov, et d'en calculer les données moyennes. Ainsi, cette approche nous permet d'obtenir des résultats dont les valeurs tendent vers les valeurs théoriques lorsque le nombre de simulations tend vers l'infini. Cet algorithme va être détaillé dans cette section.

3.1 Pourquoi l'hypothèse d'indépendance ne peut généralement pas être vérifiée

Note : (L'hypothèse d'indépendance ne peut être négligée que lorsque nos sommes dans le cas d'un faible proportion d'individus contaminés répartis dans une large population uniforme.)

L'hypothèse d'indépendance ne peut être vérifiée que lorsque nous sommes dans la situation d'une population où chaque individu est connecté à tous les autres, tel que dans la population W^{full}. En effet, la probabilité qu'un individu soit infecté va dépendre fortement de l'état infectieux de ses voisins, car par exemple, un voisin infecté peut avoir été contaminé par un autre voisin directe ou indirecte qui est peut être connecté également à un grand nombre de voisins du noeud auquel nous nous intéressons. Ainsi, en fonction de la structure du graphe représentant le population et du chemin par lequel se propage le virus, la situation réelle peut s'écarter fortement du modèle posant l'hypothèse d'indépendance.

De plus, dans l'étude de *Deepayanetal.*, le modèle utilisé n'utilise que deux états individuels possible pour les individus de la population : I et S. Ainsi, un noeud infecté qui guéri retrouve un statu S, et peut à nouveau être infecté par le virus. Contrairement à notre chaîne de Markov qui évolue vers un état final absorbant, un individu d'une population de ce modèle va être (ré-)infecté de façon cyclique. La probabilité qu'un individu donné soit infecté à un moment donné va donc, au delà d'un certain moment, rester relativement uniforme avec le temps, ce ui n'est pas le cas dans notre modèle prenant en compte un état d'immunisation.

3.2 Présentation de la classe *Simulations.java*

Afin d'étudier efficacement la propagation de la maladie dans notre population par des simulations, nous avons besoin d'une part d'un algorithme permettant de générer des "parcours" de la chaîne de Markov efficacement (l'obtention de données fiables demandant un grand nombre de simulations), et d'autre part d'un outil informatique nous permettant d'adapter facilement les simulations aux diverses mesures que l'on pourrait prendre à l'encontre de la maladie, et ce afin d'en évaluer l'efficacité.

Pour ce faire, la classe **Simulations.java** a été implémentée. Les objets instanciés à partir de cette dernière contiennent la matrice d'adjacence représentant la population étudiée, mais aussi des tableaux permettant de stocker les données issues de chaque simulation. Ces données contiennent le temps nécessaire avant disparition de la maladie (correspond à l'arrivée dans un état absorbant), ainsi que, et ce pour chaque unité de temps, le nombre d'individus de chaque statuts individuel : I, R et S.

Les simulations sont réalisées les unes après les autres par la méthode **StartSimulations()** associée à l'objet de type **Simulations**, avec la possibilité, pour chaque "intervalle" de simulations de taille désirée, de changer les paramètres de la simulation afin de les comparer entre eux.

Ainsi, pour une n^{ieme} simulation, la méthode **StartSimulations()** va se référer à l'index correspondant au numéro de l'intervalle auquel appartient la n^{ieme} simulation, dans le tableau **frame**. Elle prendra alors connaissance des paramètres à utiliser pour générer la réalisation de la chaîne.

Ce tableau contient des objets de type **SimulationsFrame** permettant à **StartSimulations** de connaître la valeur des paramètres de base tel que :

- β
- μ
- la façon de générer l'état initial : soit en sélectionnant de façon équiprobable un individu de la population à infecter en t_0, soit en sélectionnant, toujours de façon équiprobable, $0,5\%$ des individus de la population à infecter en t0

Mais grâce à ces objets, nous pouvons également faire varier la valeur de ces paramètres au cours d'une même simulation, reflétant par exemple l'impact de mesures prises à un temps déterminé et impactant la valeur de β et/ou μ. Nous pouvons également utiliser ces objets pour modéliser

une campagne de vaccination, en encodant le nombre de vaccinations à réaliser à l'unité de temps voulu, et quelle stratégie utiliser pour la sélection des individus à vacciner.

3.3 Simulations sur W^{lin} et W^{full}

Afin de répondre à la question 2.2 et 2.3, j'ai implémenté la méthode statique **StartTypeBasic()** dans la classe **SimulationsLayout.java**. Cette méthode construit et pilote un objet de type **Simulations** permettant de réaliser un nombre déterminé de simulations toutes identiques, réaliser la moyenne des données obtenues et en exporte les courbes épidémiques ainsi que le temps moyen avant disparition de la maladie dans un fichier *.csv*.

Pour répondre à la question 2.2, cette méthode est donc appelée 4 fois depuis la classe **Main.java** afin de compiler les valeurs pour les populations W^{full} et W^{lin} avec comme paramètre des valeurs $\beta = 0,5$, $\mu = 0,2$ et un état initial ne comportant qu'un seul individu infecté aléatoirement.

Les données pour chaque population sont générées une première fois à partir de la moyenne de 10 simulations (comme demandé dans l'énoncé), puis une nouvelle fois à partir d'une moyenne issue de 50 000 simulations, de façon à obtenir des données beaucoup plus stables et tendant vers le modèle exact. Elles sont exportées dans les fichiers suivants :

— $W\_lin\_basic\_10simu.csv$

— $W\_lin\_basic\_50000simu.csv$

— $W\_full\_basic\_10simu.csv$

— $W\_ful_basic\_50000simu.csv$

Ces données sont représentées dans la figure 3, présentée ci-dessous.

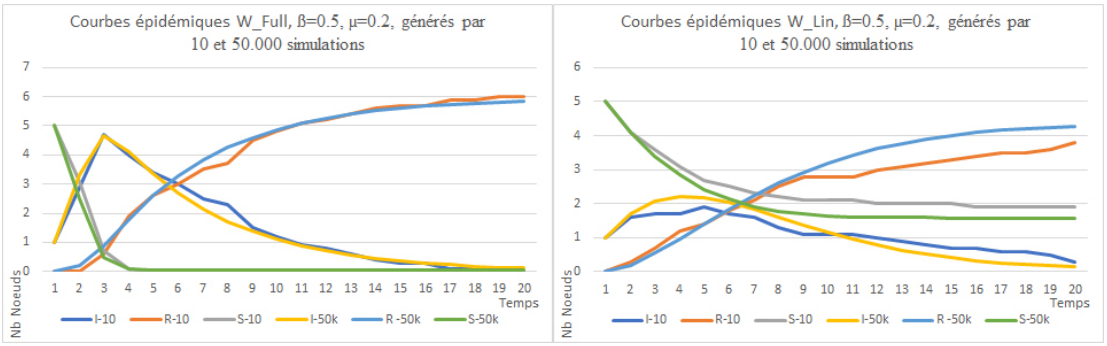

FIGURE 3

Les temps moyens avant disparition de la maladie donnés par ces simulations sont :

— 12,3 pour W^{Full} à 10 réalisations,

— 12.85366 pour W^{Full} à 500 réalisations (rappel valeurs théoriques : 12,8183),

— 15 pour W^{Lin} à 10 réalisations,

— 11,154 pour W^{Lin} à 500 réalisations (rappel valeurs théoriques : 11,6860).

Même si 10 simulations ne nous permettent pas d'obtenir des courbes très stables, ces dernières forment déjà une esquisse très proche de la situation obtenue en augmentant fortement le nombre de simulations. Cette constatation reflète la relativement faible dispersion des données générées par les simulations. C'est un élément très important, car cela signifie que les données théoriques et expérimentales que nous générons représentent fidèlement la situation réelle que

nous sommes susceptibles de rencontrer. Nous avons en effet peu de chances qu'une réalisation de la chaîne de Markov s'éloigne très fort des valeurs moyennes.

Nous remarquons également que lorsque le nombre de simulations est grand, les courbes obtenues tendent vers le modèle exact étudié précédemment. Nous pouvons donc en conclure que les simulations sont un moyen efficace d'approcher les résultats d'une situation qui serait normalement incalculable par le modèle exacte.

3.4 Simulations de base sur W^{Big}

Afin d'étudier la propagation d'une maladie virale de façon plus réaliste, nous avons à notre disposition un fichier contenant les données d'adjacence d'une population de 2000 individus.

Nous allons donc toujours utiliser la méthode **StartTypeBasic()**, mais cette fois avec la matrice d'adjacence W^{Big}, importée du fichier $Wbig\_sparse.txt$ et encodé dans un objet de type $Matrix$ au format colonnes-compressées par la méthode **matrixFromSparseile**.

Par rapport à l'exercice précédent, nous allons cette fois utiliser comme états initiaux des situations ou $0,5\%$ des individus de la population, choisis de façon équiprobable, sont infectieux en t_0.

Les résultats moyens réalisés sur 1000 simulations sont présentés dans la figure 4 représentée ci-dessous. Le temps moyen avant disparition de la maladie au cours de ces simulations est de 42.956 unités de temps.

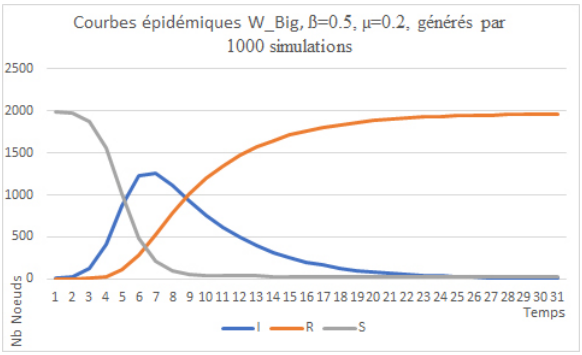

FIGURE 4

Il est intéressant de constater que les résultats obtenus par simulations sur le modèle W^{Big} se rapprochent fortement, toutes proportions gardées, des données obtenues lors de l'étude de W^{Full} avec un pic épidémique très rapide et élevé comprenant plus de la moitié de la population en état infectieux simultanément, et des états finaux ou presque l'intégralité des individus de la population ont contracté la maladie au cours de l'épidémie (et donc ou presque tous les individus sont "R").

Pourtant, dans la population W^{big}, les individus sont en moyenne connectés à une très faible portion du reste de la population.

Pour comprendre cela, il faut s'intéresser aux particularités du graphe représenté par W^{big}. Ce graphe, fait pour représenter la population, est ce que l'on appel un "scale-free network". Cela signifie que la distribution des degrés des noeuds le composant suit une loi de puissance.

Ainsi, une très faible proportion des individus de la populations sont ce que l'on appel des "hubs", et sont connectés à énormément d'autres personnes alors que la grande majorité des individus de la population sont connectés à un petit nombre d'autres individus. Ainsi, malgré le faible degré moyen de la population, qui semble à première vue la rapprocher de la situation de

la population W^{Lin}, la présence de ces "hubs" va détériorer drastiquement la contagion au sein de W^{Big} en éparpillant rapidement le virus.

Dans le cas réel de la lutte contre le COVID-19, ces hubs sont des personnes qui, pour une raison ou une autre, sont en contact avec beaucoup d'autres, tel que des hommes politique, ou de façon plus courante des épiciers ou autres vendeurs. C'est ainsi que l'une des théories les plus citée sur l'origine de l'éparpillement du COVID-19 semble être un marché de la ville de Wuhan en Chine, ou un individu (peut être même le patient zéro) aurait joué le rôle de "hub" en contaminant un grand nombre d'autres personnes rapidement.

3.5 Simulations prenant en compte la prise de mesures contre la propagation de l'épidémie.

Dans cette section, nous allons dans un premier temps voir comment utiliser le code que j'ai rédigé afin d'évaluer d'abord séparément, puis conjointement, l'impact de la modification des paramètres de base et , ainsi que l'impact d'une campagne de vaccination, et enfin comment générer des simulations personnalisées regroupant chronologiquement les diverses mesures que l'on pourrait réellement prendre afin de voir leur impact sur les courbes épidémiques.

3.5.1 Étude de l'influence de β.

Tout comportement permettant de réduire la probabilité de transmettre le virus d'un individu à un autre tel que décrit dans la question 2.4.a revient à influencer négativement la valeur de β.

La classe **SimulationsLayout.java** a donc été implémentée. Cette dernière contient la méthode **StartTypeBetaVar()** destinée à générer des données permettant de visualiser l'impact d'une modification de la valeur de β sur les courbes épidémiques.

Cette méthode va donc générer et exécuter un objet de type **Simulations** permettant de générer les courbes épidémiques avec une variation déterminée de la valeur de β pour chaque intervalle. Dans notre cas, nous réalisons un **Frame** contenant une valeur de μ fixée à 0.2 et une valeur de β allant de 0.1 et 0.6 avec un augmentation de 0.01 par intervalle. Les données générées sont des moyennes réalisées sur 100 réalisations par intervalle. Les données sont exportées dans le fichier $W\_big\_Beta\_var\_mu02.csv$ et sont représentées dans les graphes de la figure 5 affichée ci-dessous.

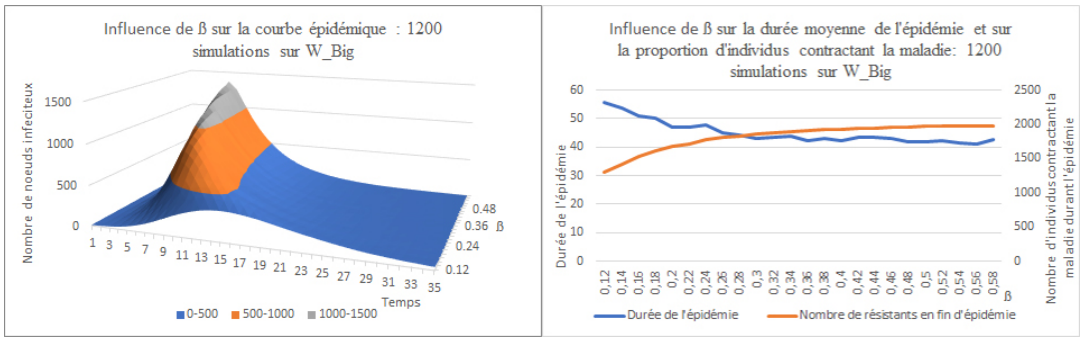

FIGURE 5

Ces données nous permettent de nous rendre compte qu'une diminution de la valeur du paramètre β a un effet d'étalement sur la courbe épidémique. Étaler cette courbe peut se révéler très important, car cela va avoir comme effet de diminuer la proportion de la population simultanément malade. Or, suivant la situation, ces individus malades peuvent nécessiter des ressources dont la disponibilité est réduite, tel que des places en institutions hospitalières ou

des médicaments dont les stocks et la production sont limités et/ou doit augmenter. Il est donc nécessaires de prendre les mesures nécessaires afin de ne pas surcharger notre capacité de soins.

Le graphique de gauche nous montre également que pour obtenir un aplatissement significatif de la courbe épidémique, la diminution de β doit être très forte, ce qui sous-entend que les mesures prises doivent être strictes et fortes.

Un autre aspect que nous pouvons remarquer, est que l'abaissement de la courbe épidémique en agissant sur β va se faire globalement au détriment de la durée de l'épidémie. En effet, cela va avoir comme conséquence de retarder le temps nécessaire à la disparition de la maladie, ce qui peut être important à prendre en compte, principalement du point de vue des conséquences économiques et sociales engendrées.

Dernièrement, nous remarquons également que la diminution de β, permet de diminuer la proportion de personnes ayant un statut "Résistant" en fin d'épidémie, et donc qui ont contracté la maladie au cours de l'épidémie.

Cet effet peut se révéler très important. En effet, même si le plus important peut à première vue être d'étaler la courbe épidémique, il ne faut pas perdre d'esprit que même sans surcharge, nos institutions hospitalières peuvent, suivant l'agent infectieux, être d'une relative inefficacité pour réduire le taux de mortalité dû à la maladie. Dans ce cas, la stratégie la plus importante consistera à diminuer le plus possible la proportion de la population contractant le virus au cours de l'épidémie, car cette proportion est alors directement liée au taux de mortalité.

3.5.2 Étude de l'influence de μ.

Traiter les patients avec un médicament permettant d'accélérer la guérison (ou du moins la durée d'excrétion virale) tel que décrit dans le point d de l'énoncé, revient à augmenter la valeur de μ. Cela est en pratique souvent difficile, voir impossible, dans le cadre de maladies virales. Cependant, certains traitements ou découvertes fortuites peuvent parfois montrer un potentiel effet sur la durée d'excrétion. Ainsi, dans le cadre de la lutte contre le COVID-19, certains médecins semblent apporter des débuts de preuves d'efficacité de molécules tel que des antirétroviraux, l'hydroxychloroquine ou le fipronil.

A côté de ces traitements, qui ne vont généralement avoir qu'une influence limitée sur μ, il faut ajouter l'impact d'un dépistage et d'une mise en quarantaine rapide des individus excréteurs. En effet, nous pouvons considérer des individus placés en quarantaine comme étant guéris d'un point de vue épidémiologique, étant donné qu'ils ne peuvent plus transmettre la maladie. Ainsi ce sont ces mesures qui peuvent avoir le plus d'influence sur la durée d'excrétion possible du virus.

De la même façon que pour le paramètre β, la méthode **startTpeMuVar()** de la classe **SimulationsLayout.java** permet de générer le fichier $W\_big\_Mu\_var\_beta05.csv$. Les données exportées nous permettent de générer la figure 6 affichée ci-dessous.

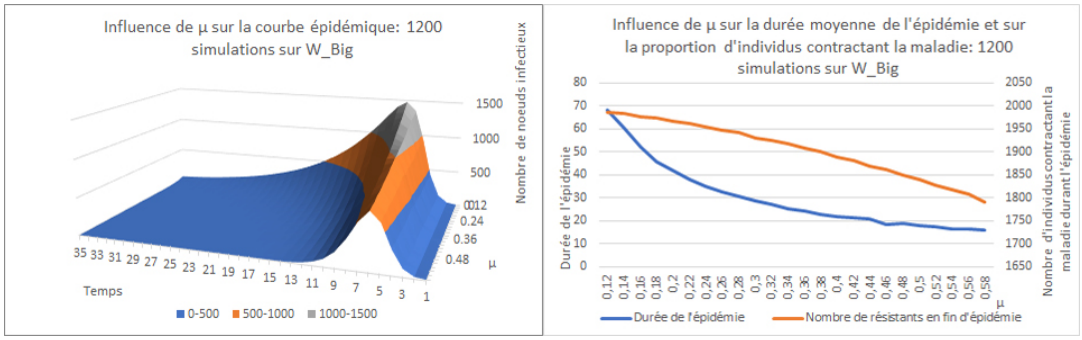

FIGURE 6

La diminution de la durée de la période de contagion (et donc l'augmentation de la valeur du paramètre μ), permet, tout comme les mesures agissant sur β, de réduire la hauteur de la courbe épidémie et la proportion de personne contractant la maladie au cours de l'épidémie. Cependant, la grosse différence avec le cas précédent est que cela ne s'accompagne cette fois pas d'un allongement de la durée de l'épidémie, mais bien d'une diminution de cette dernière.

L'amélioration de la valeur de μ lorsque $\beta = 0.5$ engendre relativement peu d'amélioration sur la proportion d'individus de la population contractant la maladie au cours de l'épidémie. Nous verrons plus loin que la situation n'est plus la même lorsque la valeur de β est plus faible.

3.5.3 Étude conjointe de β et de μ

Nous avons vu que pour obtenir un effet significatif sur l'épidémie, une action isolée d'un seul paramètre ne permet généralement que peu d'amélioration et demande une modification importante de sa valeur, qui n'est généralement pas possible dans la pratique. Une solution qui apparaît comme évidente serait donc d'attaquer sur plusieurs fronts afin d'améliorer conjointement plusieurs paramètres.

La méthode **startTypeBetaMuVar()** a donc été implémentée dans la classe **SimulationsLayout.java**. Elle combine les actions des deux méthodes utilisées précédemment. Les données générées sont issues des moyennes réalisées sur 100 simulations par couples de valeurs β et μ et sont exportées dans le fichier $W\_big\_MuAndBeta\_Var.csv$.

Elles permettent entre autre de générer les courbes présentées dans la figure 7 présentée ci-dessous :

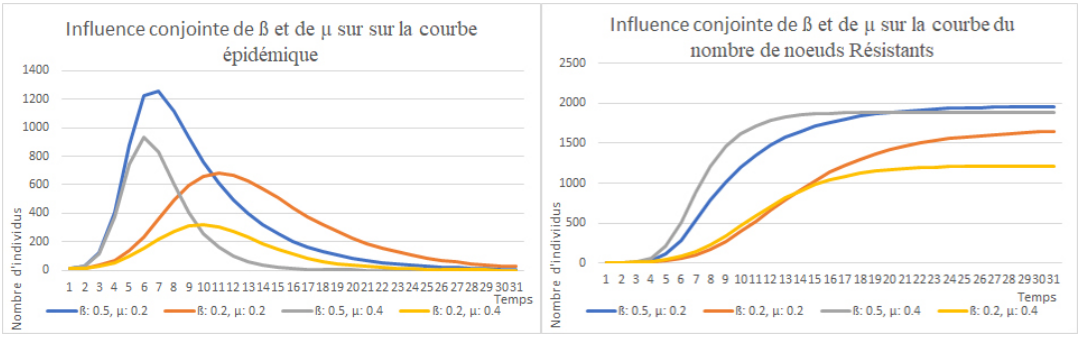

FIGURE 7

Ces deux graphes nous permettent d'une part de mieux mettre en évidence les effets respectifs d'une amélioration de β et μ sur les courbes épidémiques, mais également de mettre en évidence une réelle synergie entre eux. Ainsi, l'effet d'étalement de la courbe par une amélioration de β (rsp. μ) sera d'autant plus forte que l'on agit également sur μ (rsp. β).

D'autre part, il est important de remarquer que, alors que doubler la valeur de μ seule semble avoir peu d'effet sur la proportion de la population contractant la maladie au cours de l'épidémie (voir même un effet négatif), il n'en est pas de même lorsque l'on améliore conjointement la valeur de β (en la divisant par 2).

Nous pouvons générer les graphiques de la figure 8 afin de mieux évaluer l'impact sur la durée de l'épidémie et sur la proportion de la population contractant la maladie :

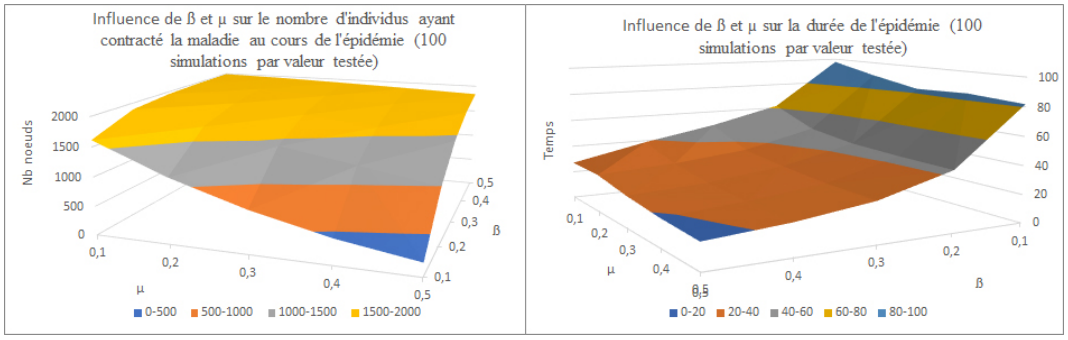

FIGURE 8

Concernant l'influence de mesures conjointes sur β et μ, les graphes de la figure 8 confirment les observations effectuées précédemment, et nous montrent que l'amélioration de μ permet de compenser partiellement l'effet néfaste de celle de β sur la durée de l'épidémie.

Le plus intéressant est surtout l'impact très important que l'on peut avoir sur le nombre d'individus contractant la maladie au cours de l'épidémie, représenté par la première figure. Cette valeur étant dans beaucoup de cas directement reliée au taux de mortalité de l'épidémie, cela nous montre qu'une amélioration drastique et conjointe de ces deux paramètres permet d'éviter la grande majorité des éventuels décès.

3.5.4 Influence d'un programme de vaccination

Afin d'étudier l'influence d'une campagne de vaccination sur une épidémie, j'ai implémenté la méthode **startTypeVaccVar()** dans la classe **SimulationsLayout**. Elle fonctionne de la même façon que les autres types de simulations mais permet ici de générer des données permettant de comparer l'effet de campagnes de vaccination réalisées à un moment précis, avec un type de vaccination donné, et en fonction du nombre d'individus de la population vaccinés.

Deux types de campagne de vaccination sont envisagées :

— Un premier sélectionnant les individus à vacciner au hasard dans la population

— Un deuxième vaccinant en priorité les noeuds de plus haut degrés, c'est à dire les individus étant en contact avec le plus d'autres noeuds, et donc le plus susceptible de contaminer un grand nombre d'autres individus.

Si nous réalisons des simulations en vaccinant la population dès le début de l'épidémie, les données générées nous permettent, entre autre, de générer les graphes de la figure 9 présentée ci-dessous :

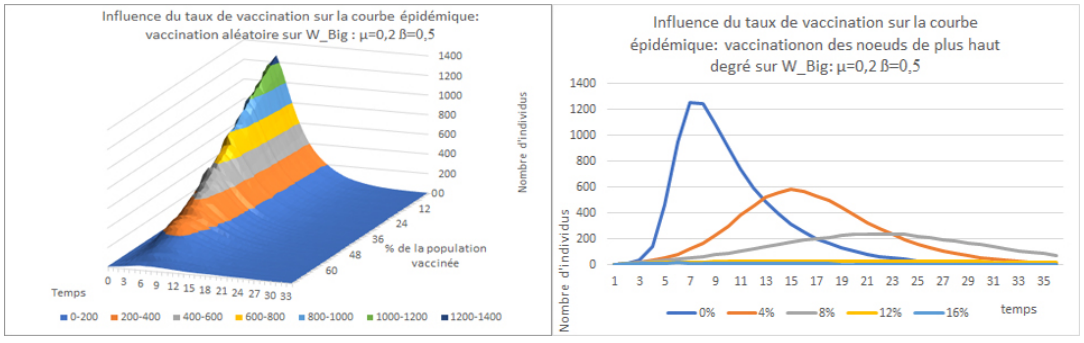

FIGURE 9

Sur la Figure 9, le graphe de gauche nous fait remarquer que l'étalement de la courbe épidémique suit linéairement la proportion d'individus vaccinés au début de la simulation. Ce graphe nous montre qu'une très importante proportion de la population doit être vaccinée afin d'obtenir un effet satisfaisant sur l'épidémie, et la presque totalité de la population doit l'être afin de mettre un terme à l'épidémie (et donc d'éviter de nouvelles contaminations).

Cependant, dans une situation réelle, la production d'un nouveau vaccin prend du temps et il est inconcevable de pouvoir vacciner l'intégralité de la population dès sa mise sur le marché. Il est donc nécessaire de savoir sélectionner qui vacciner en premier pour avoir le plus rapidement possible un impact significatif sur la propagation de la maladie.

La propriété principale des "scale-free network" tel que l'on représente notre population, est le fait que la distribution des degrés des noeuds qui la composent suit une loi de puissance. Il paraît donc évident qu'une très petite minorité des individus de la population vont avoir un pouvoir contaminant largement supérieur aux autres, et c'est donc ces individus qu'il va être intéressant de sélectionner. Si nous nous intéressons au graphe de droite de la Figure 9, nous remarquons l'effet fulgurant que la vaccination peut permettre sur la courbe épidémique, lorsque l'on vaccine les quelques % d'individus les plus connectés.

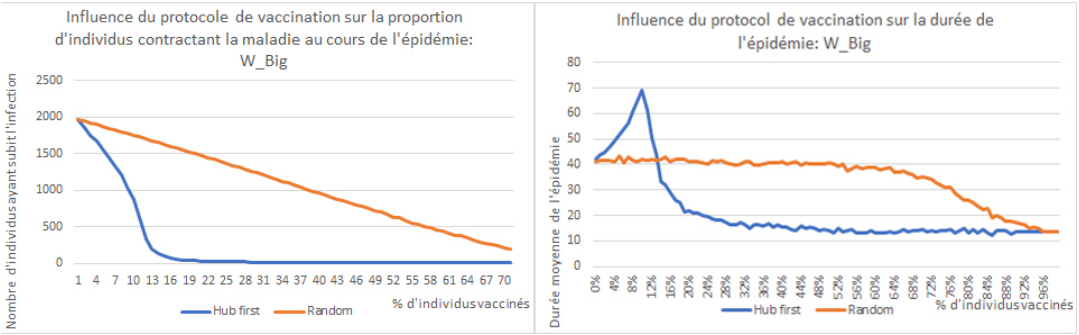

FIGURE 10

Les deux graphes de la Figure 10, également issus des données exportées précédemment, nous permettent de mettre en évidence plusieurs informations intéressantes.

Pour commencer, la vaccination aléatoire ne va provoquer qu'une diminution relativement linéaire de la proportion de la population qui va contracter le virus au cours de l'épidémie. Les individus n'étant pas infectés au cours de l'épidémie étant en fait limité aux individus traités lors de la campagne de vaccination. De plus, le graphe de droite de la figure 10 nous montre que malgré cette diminution, la vaccination ne va avoir que peu d'impact sur la durée de l'épidémie, la campagne aléatoire ralentissant cette dernière sans pour autant la stopper (à mois de vacciner une très grande proportion de la population.

Par contre, lorsque l'on oriente la vaccination vers les individus les plus connectés, il est intéressant de voir que l'impact sur la proportion de la population contractant la maladie au cours de l'épidémie s'en retrouve très rapidement impacté, et est la conséquence d'une disparition très rapide de la maladie.

Il est par ailleurs intéressant de remarquer que jusqu'aux 10% d'individus les plus connectés vacciné, le protocole va se traduire par un allongement de la durée de l'épidémie. La présence d'individus vaccinés parmi les plus connectés la ralentissant, mais ne permettant pas encore d'isoler totalement les foyers et donc de l'arrêter. Par contre, lorsque l'on dépasse un certain seuil, la maladie disparaît brutalement.

Nous pouvons ainsi introduire une nouvelle notion, un seuil d'immunité globale représentant la proportion minimale d'individus à vacciner dans une population donnée pour empêcher la

"percollation" du virus en son sein, et donc la propagation de la maladie.

Si l'on regarde les données relatives à la population W^{Big} lorsque la campagne de vaccination est réalisée aléatoirement en t_0 et que 10% de la population est contaminée aléatoirement en t_0, nous remarquons qu'il faut vacciner en moyenne 95% des individus pour cesser la contamination de nouveaux noeuds. Si l'on réalise cette fois cette campagne de vaccination en sélectionnant les individus ayant le plus de relations, il ne faut plus que 30% de vaccinés avant de stopper totalement la propagation la maladie.

Dans un contexte réel, il faut, outre les problèmes logistiques que représentent une mise sur le marché rapide et à grande échelle d'un vaccin, prendre en compte le fait qu'à de rares exceptions près, un vaccin n'est généralement pas utilisable en curatif, et d'autre part que l'immunisation individuelle procurée par la vaccination n'est pas instantanée. Deux semaines sont généralement nécessaires afin d'obtenir une production d'anticorps suffisante pour procurer une réelle protection contre la maladie. Il faut aussi prendre en compte le fait que certains agents viraux disposent de mécanismes d'immuno-évasion très efficaces leur permettant d'être invisibles vis à vis de nos mécanismes d'immunité acquise, et rendant par la même occasion toute vaccination impossible. De plus, il faut également signaler que tout agent viral, et principalement les virus à ARN tel que le COVID-19, sont sujet à un fort pouvoir mutagène. Un vaccin peut ainsi du jour au lendemain se retrouver partiellement ou totalement inefficace, de même que l'immunité acquise des individus ayant déjà contracté le virus, et qui pourraient ainsi déclarer à nouveau la maladie.

3.5.5 Modéliser des situations

Comme déjà mentionné plus haut, la lutte contre une épidémie est multi-factorielle, et il est important d'avoir un outil informatique performant afin de prédire son évolution et quantifier l'effet des divers mesures que l'on peut prendre. En effet, beaucoup de ces mesures risquent d'avoir des coût conséquents, que ce soit d'un point de vue économique ou humain. Les simulations permettent donc d'évaluer le mieux possible l'effet de ces divers mesures sur l'épidémie, et ce afin de réaliser une étude bénéfice/risque la plus précise possible permettant de juger de leur pertinence, et prendre les meilleures décisions.

Les structures de données implémentées dans ce projet ont expressément été créées de façon à pouvoir générer des simulations dynamiques. C'est à dire, pour lesquelles nous pouvons facilement programmer l'évolution des paramètres de base (β et μ) au cours d'une même simulation, ou encore programmer des actions, tel que des protocoles de vaccination, sur une partie de la population.

Par exemple, la méthode **startSimuPersoA()** de la classe **SimulationsLayout** montre comment encoder un **SimulationFrame** de façon à représenter les diverses mesures prises au cours du temps.

Dans cette exemple nous représentons la situation suivante :

— t_0 : Début de l'épidémie dans la population étudiée. Nous partons d'une situation initiale comportant 0.5% de la population infecté de façon aléatoire en t_0 (état initial de type 1). Les paramètres de base sont : $\beta = 0.5$ et $\mu = 0.2$.

— t_1 : Face à la situation dans des pays où le virus a été introduit plus tôt et devant la dangerosité manifeste de ce dernier, le gouvernement prend rapidement des mesures de confinement. La valeur du paramètre β est multipliée par un facteur 0.2 pour descendre à une valeur de 0.1.

— t_8 : Des progrès dans les tests de dépistage et la découverte de divers traitement agissant sur la durée d'excrétion virale permettent d'augmenter la valeur de μ. Cette dernière est multipliée par un facteur 1.2.

— t_{12} : Face aux difficultés économiques rencontrées par les entreprises, et rassurées par la baisse de la courbe épidémique, les autorités initient le dé-confinement. Malgré l'amélioration des mesures d'hygiène et de distanciation sociale, le comportement irresponsable de certaines personnes ramènent le paramètre β à sa valeur initiale.

— t_{16} : les universités et les laboratoires s'adaptent et mettent en place de nouvelles techniques de diagnostique. Le dépistage se fait donc à plus grande échelle et la mise en quarantaine des individus infectieux est plus précoce. La valeur de μ est une nouvelle fois multipliée par un facteur 1.2. t_{20} : Une firme pharmaceutique parvient à mettre sur le marché un premier vaccin. Sa production étant dans un premier temps fort limitée, la vaccination se fait de façon sélective sur les 5% de la population n'ayant pas été contaminés et étant les plus à risque (les noeuds de plus haut degrés).

Les résultats moyens de 3000 simulations avec ce **SimulationFrame** peuvent être synthétisés dans la Figure 11 présentée ci-dessous. Le temps moyen avant disparition de la maladie est de 39,785 unités de temps.

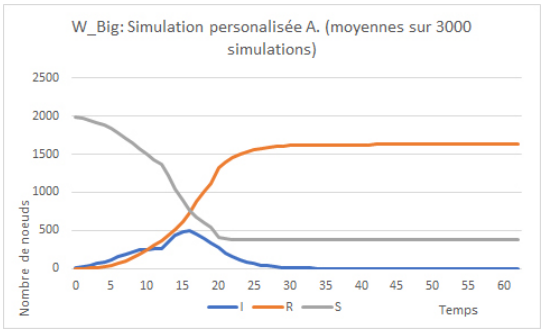

FIGURE 11

Une infinité de combinaisons est envisageable. Dans l'exemple choisi, nous remarquons que le dé-confinement précoce précède une forte ré-augmentation des cas dans la population étudiée.

Afin de déterminer une meilleure stratégie de dé-confinement, nous avons généré, de la même façon, les résultats présentés dans la Figure 12 :

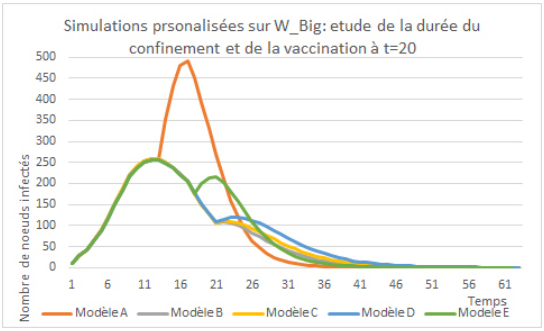

FIGURE 12

— Le modèle A : dé-confinement à t_{12} et vaccination sélective à t_{20}.

— Le modèle B : dé-confinement à t_{20} et vaccination sélective à t_{20}.

— Le modèle C : dé-confinement à t_{20} et vaccination non-sélective à t_{20}.

— Le modèle C : dé-confinement à t_{20} et pas de campagne de vaccination.

— Le modèle E : dé-confinement à t_{17} et vaccination sélective à t_{20}.

Les constatations que l'on peut faire sont assez évidentes :

Un dé-confinement à t_{12} serait une catastrophe : la courbe épidémique reprendrait une forte croissance instantanément.

Prolonger le confinement jusqu'à t_{20} par contre permet d'éviter un deuxième pic. En effet, même en l'absence de vaccination, le nombre d'individus infectieux va légèrement augmenter pour finalement décroître. Si l'on associe le dé-confinement en t_{20} à un protocole de vaccination sur 0.5% de la population, nous évitons même cette légère croissance.

Nous remarquons également qu'à t_{20}, on a peu de différence entre l'effet d'une vaccination sélective (en sélectionnant les individus ayant le plus de relations) et une vaccination en sélectionnant des individus sains aléatoirement. Cela peut s'expliquer du fait que notre population est un réseau sans échelle. En effet, les noeuds de degrés élevés ont déjà presque tous été contaminés en t_{20}. Une vaccination sélective se justifierait donc plutôt dans le cadre d'une vaccination plus tôt durant l'épidémie, sur la population W^{Big}.

D'ailleurs, ces individus "hubs" ne pouvant à ce moment là déjà plus contracter la maladie, et la transmettre aux autres, un protocole vaccinal aléatoire a lui aussi tendance à devenir négligeable, la propagation de la maladie étant déjà en fort déclin.

Une situation intermédiaire envisageable, dans le cas de notre exemple, serait un dé-confinement en t_{17}. Il s'accompagne bien d'une légère ré-augmentation des cas, mais ne dépassant pas le précédent pic.

4 Conclusion

La prise de mesures contre une propagation exponentielle du COVID-19 semble être primordiale afin d'éviter l'hécatombe que représenterai la non-action et une immunisation passive de la population. Cependant, prendre des mesures de façon anarchiques pourrait également être une tragédie de part l'impact économique et social qu'elles peuvent générer sans pour autant avoir de réels effets bénéfiques. Il est donc crucial d'utiliser des modèles mathématiques les plus performants possibles afin de réaliser des simulations fidèles permettant de juger de la pertinence d'une mesure, ainsi que la façon optimale de la mettre en place : à quel moment, pendant combien de temps, et avec quelle intensité.

Le modèle créé dans ce travail reste encore beaucoup trop limité pour être appliqué dans des conditions réelles. Énormément de choses ne sont pas encore prises en compte, tel que la durée d'incubation de la maladie, le taux d'individus asymptomatiques, et bien d'autre. De plus, la modélisation de la propagation de la maladie à partir de valeurs données des paramètres de base reste une très petite part du travail. Le plus compliqué reste de déterminer la valeur de ces paramètres, la façon dont ils vont évoluer, et la façon dont les mesures prises vont influencer ces valeurs.

De plus, la population W^{Big} utilisée pour les simulations est loin de représenter la population réelle et est très petite. L'utilisation des scale-free network semble plus adaptée à la représentation de réseaux informatiques ou de réseaux sociaux plutôt que les réelles contacts physiques que les individus d'une population peuvent avoir entre eux. Une meilleure représentation, d'autant plus que nous sommes dans une situation où les individus essaient de limiter les contacts physiques entre eux, serait une distribution plus uniforme des degrés des noeuds, avec la présence de nombreux hubs locaux, plus petits, et qui représentent les commerces alimentaire de proximité ou les transports en communs.