Université de Montréal

Dynamique épidémique

Mathis Farah-Lajoie (20280102)
MathisFarah

Maxim Moreau (20269875)
Max80780

Myriam Menia (20281484)
Myriam7865

Travail présenté à Timothée Poisot dans le cadre du cours
BIO 2045 « Simuler le Vivant »

Source du projet

2026-04-08

Table des matières

Introduction 1
Mise en contexte 1
Questions 1
Hypothèses et résultats attendus 1
Présentation du modèle (Contraintes du modèle) 1
Structure et dynamique épidémique 1
Vaccination et tests de dépistage 2
Contraintes budgétaire et d'intervention 2
Stratégie de dépistage et vaccination 2
Modifications du modèle (justification biologique) 3
Implémentation 3
Packages nécessaires 3
Inclure du code 3
Simulations 12
Création de nouvelles fonctions 12
Paramètres initiaux 13
Simulation 106
Analyse des résultats 110
Présentation des résultats 112
Discussion 112
Bibliographie 112

Introduction

Mise en contexte

Alors, les maladies infectieuses peuvent se propager rapidement dans une population, surtout quand celle-ci ne possède pas une immunité déjà présente et est donc naive [1]. Cela peut même mener à un nombre très élevé de décès, surtout lorsque la maladie a beaucoup d'effets néfastes et est donc grâve et virulente [2]. La vitesse de propagation de celle-ci dépend de divers variables comme le taux d'infection, la durée de la maladie chez un individu et des contacts entre individus [3]. Il y a cependant des façons de contrer des épidémies, comme la vaccination et le dépistage, qui permettent de réduire le nombre d'individus susceptibles d'être malades et permettent d'identifier les personnes infectieuses afin d'intervenir de manière plus précise, et ce même sans symptômes appararents chez les individus [4]. En contre partie, ces stratégies sont souvent limitées par des contraintes, comme un budget [5], ce qui force parfoit de faire le choix entre tester davantage ou vacciner plus d'individus. C'est donc dans ce contexte que les modèles de simulation deviennent utiles pour comparer différentes stratégies et mieux comprendre leur impact sur la propagation d'une maladie [3].

Questions

Dans le modèle de dynamique épidémique de propagation d'une maladie infectieuses dans une population qui est naive, quelle stratégie, qui combine dépistage et vaccination, permet de minimaliser la mortalité dans la population tout en respectant une contrainte de budget limité à 21 000$? De plus, comment cette stratégie se compare à une situation où il y a absence d'interventation?

Hypothèses et résultats attendus

Donc, pour émettre certaines hypothèses, nous nous attendons qu'une stratégie de vaccination et dépistage va nécessairement réduire la mortalité par rapport à l'absence d'intervention. Comme les individus infectieux sont asymptomatiques, il va falloir cibler une stratégie où les tests de dépistage vont permettre de mieux cibler la vaccination et donc, d'être plus efficaces qu'une vaccination aléatoire seule sans dépistage pour ne pas dépasser notre budget.

Alors, l'efficacité de la stratégie va surtout dépendre de l'utilisation du budget qui est limité, ce qui implique donc d'avoir un certain compromis entre le nombre de tests et de vaccins. Il faut aussi garder en tête la variabilité des simulations vu que l'intervention débute seulement qu'après le premier décès et que le vaccin n'est actif qu'après deux jours, donc une partie de la propagation devrait tout de même se faire de façons aléatoire.

On s'attend donc à une diminution de la mortalité grâce à notre stratégie, mais cela avec une certaine variabilité entre les simulations vu la stochasticité du modèle.

Présentation du modèle (Contraintes du modèle)

Structure et dynamique épidémique

Alors, le modèle simule la dynamique épidémique de la propagation d'une maladie infectieuse dans une population de 3750 individus qui sont initialement naïfs, donc sans aucune immunité. Les individus sont répartis sur une lattice de taille $-50, 50$ sur les deux dimensions, il y aura donc des interactions entre voisins pour la propagation de la maladie. Donc, la transmission de la maladie dépend d'un taux d'infection fixé à des probabilité de devenir infectieux par contact

de 0,4 lors des contacts entre individus sains et infectieux. Une fois infecté, un individu reste infectieux pendant 21 jours qui finit toujours par le tuer au bout de ces 21 jours. Alors, un taux d'infection relativement élevé et une durée de l'infection durant longtemps peuvent être typiques de maladies infectieuses fortement transmissibles. Cela peut alors entrainer une propagation rapide de la maladie dans une population qui est naive entre gens infectieux et ceux susceptibles de l'être par contact [1].

Vaccination et tests de dépistage

Pour ce qui est du vaccin, celui-ci est entièrement efficace et donc les individus vaccinés ne peuvent plus être infectés et transmettent la maladie. Cependant, le vaccin n'est actif que deux jours après son administration, il y a donc un délai dans la protection. Cela représente le temps nécessaire au développement d'une réponse immunitaire après l'inoculation du vaccin, malgré que l'efficacité absolue de celui-ci est une simplification des vaccins qui ne sont jamais parfaits, mais souvent proche de l'être (ex:95%) [6]. Les individus infectieux sont asymptomatiques, comme il est possible pour certaines maladies [4], donc il faut utiliser des tests de dépistage antigéniqes rapides pour les détecter. Ces test ont une efficacité de 95%, mais ils ne permettent pas de connaître depuis combien de temps un individu est infecté. Cela représente bien le fait que des tests antigéniqes rapides peuvent être imparfaits vu par exemple des faux négatifs [7]. On ne peut donc connaître la prévalence de la maladie dans la population autrement que par des tests.

Contraintes budgétaire et d'intervention

Finalement, les interventions pour contrer la propagation de l'épidémie sont resstreintes par un budget total de 21 000, $chaque vaccin coûte 17$ et chaque test de dépistage coûte 4$. Une fois le budget épuisé, il n'est plus possible de tester ou de vacciner. Cela est courant en santé publique d'avoir à faire avec un budget limité ce qui limite les ressources pour les mesures d'interventions pour contrer l'épidémie. Il faut donc optimiser les ressources disponibles pour limiter la propagation de la maladie et limité la mortalité [5]. Finalement, les interventions pour contrer l'épidémie comme la vaccination et les tests ne peuvent commencer qu'après l'apparition du premier décès dans la population. Cela représente le fait qu'il existe un délai entre la détection des premiers cas et le début des mesures d'interventions contre l'épidémie [8].

Stratégie de dépistage et vaccination

Alors, l'intervention va commencer dès qu'un premier individu meurt dans la population, ce qui indique alors que la maladie est déjà présente dans la population. À partir de ce moment, on effectue des tests RAT dans un rayon de 21 cellules autour de cet individu. Cela en raison que le rayon correspond à la durée de la maladie durant laquelle un individu peut infecté un autre individu, donc le rayon représente la zone où la maladie a le plus de chances de s'être propagée.

Par la suite, les individus qui sont testés positifs dans ce rayon vont être vaccinés. Même si le vaccin est actif qu'après deux générations, cela permet quand même de limiter rapidement la transmission de la maladie vu qu'on empêche les individus vaccinés de devenir ou de rester infectieux. Ce même processus est répété à chaque fois qu'un nouvel individu décède, on va alors tester encore dans un rayon de 21 à chaque fois puis vacciner les positifs. Cela permet donc de suivre la propagation de la maladie et de concentrer les efforts de tests et vaccinations dans les zones qui sont plus à risques, tout cela en évitant le plus possible de tester inutilement l'ensemble de la population et de dépasser le budget.

Modifications du modèle (justification biologique)

Alors, pour mettre en place la stratégie, il a fallu modifié certaines choses du modèle pour intégrer des mécanismes biologiques supplémentaires. Un état vacciné a été ajouté aux agents, qui permet de représenter les individus protégés contre l'infection après le délai de deux générations. Ce délai correspond au temps qui faut pour développer une réponse immunitaire, après cela il ne peut plus transmettre et être infecté par la maladie.

Par la suite, un mécanisme est également ajouté afin de détecter les individus infectieux asymptomatiques. Vu qu'ils ne présentent aucun symptômes, le test est le seul moyen de les identifier. L'efficacité du test (95 %) introduit une petite incertitude, ce qui représente la présence de faux négatifs et de faux positifs en conditions réelles. De plus, la propagation de la maladie se fait via des contacts directs entre individus partageant la même position, donc cela justifie l'utilisation d'un dépistage ciblé dans un rayon donné de 21 autour des individus morts. Les individus testés positifs dans ce rayon de 21 vont être vaccinés, donc leur état va changer, pour qu'ils ne puissent plus infectés ou mourir de la maladie après 2 jours. Le rayon de 21 est le temps de 21 jours avant qu'un individu décède de la maladie, donc le déplacement qu'il peut faire avant qu'il meurt où il peut propager la maladie à d'autres. Cela correspond à un délai plausible pour qu'une maladie infectieuse et sévère cause la mort des gens infectés.

Implémentation

Packages nécessaires

Initialisation du générateur aléatoire pour garantir la reproductibilité des simulations

```
import Random
Random.seed!(123456)
```

```
Random.TaskLocalRNG()
```

Librairie utilisée pour la visualisation des résultats (graphiques)

```
using CairoMakie
CairoMakie.activate!(px_per_unit=6.0)
using StatsBase
```

Puisque nous allons identifier des agents, nous utiliserons des UUIDs pour leur donner un indentifiant unique:

```
import UUIDs
UUIDs.uuid4()
```

```
Base.UUID("d166a39c-6c78-4ce3-9ba7-83f956db114f")
```

Inclure du code

Type représentant une population d'individus

```
"""
Représente un individu dans la simulation.

Attributs :
- x, y : position de l'individu sur la grille
- infectiousclock : nombre de jours restants avant la mort
- vaccinationclock : temps de latence avant que le vaccin fasse effet lorsque vacciné
- infectious : indique si l'individu est infecté et contagieux
- vaccinated : indique si l'individu est vacciné et si son vaccin fait effet
- tested : indique si l'individu est testé lors des RAT pour éviter de les tester plusieurs
fois
- id : identifiant unique

Biologiquement, un individu infecté voit son état se dégrader
jusqu'à atteindre 0 (mort).
"""

Base.@kwdef mutable struct Agent
    x::Int64 = 0
    y::Int64 = 0
    infectionclock::Int64 = 21
    vaccinationclock::Int64 = 0
    infectious::Bool = false
    vaccinated::Bool = false
    tested::Bool = false
    id::UUIDs.UUID = UUIDs.uuid4()
end
```

```
Main.var"##277".Agent
```

La deuxième structure dont nous aurons besoin est un paysage, qui est défini par les coordonnées min/max sur les axes x et y:

```
"""
Définit l'environnement spatial de la simulation.

Attributs :
- xmin, xmax : limites horizontales
- ymin, ymax : limites verticales

Les individus se déplacent à l'intérieur de cet espace,
ce qui influence la propagation de la maladie.
"""

Base.@kwdef mutable struct Landscape
    xmin::Int64 = -25
    xmax::Int64 = 25
    ymin::Int64 = -25
    ymax::Int64 = 25
end

"""
Déplace un individu aléatoirement dans la grille.

L'individu peut se déplacer d'une case dans chaque direction.
Si "torus" est activé, les bords sont connectés.

Biologiquement cela représente les déplacements des individus,
qui permettent les contacts et la transmission de la maladie.
"""
function move!(A::Agent, L::Landscape; torus=true)
    A.x += rand(-1:1)
```

```
    A.y += rand(-1:1)
    if torus
        A.y = A.y < L.ymin ? L.ymax : A.y
        A.x = A.x < L.xmin ? L.xmax : A.x
        A.y = A.y > L.ymax ? L.ymin : A.y
        A.x = A.x > L.xmax ? L.xmin : A.x
    else
        A.y = A.y < L.ymin ? L.ymin : A.y
        A.x = A.x < L.xmin ? L.xmin : A.x
        A.y = A.y > L.ymax ? L.ymax : A.y
        A.x = A.x > L.xmax ? L.xmax : A.x
    end
    return A
end

"""
Crée une population de n individus placés aléatoirement
dans le paysage.
"""
const Population = Vector{Agent}

"""
    function Population(L::Landscape, n::Integer)

Fonction qui crée une population d'agent de nombre 'n' et les places aléatoirement dans une
paysage choisi

L : Landscape, donc le paysage (lattice d'une certaine dimension)
n : nombre d'agent qu'on veut avoir dans la population
"""
function Population(L::Landscape, n::Integer)
    return rand(Agent, L, n)
end

"""
    function figureEtatSelonTemps(S, I, V)

Figure permettant d'observer le nombre de différents états des agents à travers le temps

S : vecteur contenant le nombre d'agent sains pour chaque tick
I : vecteur contenant le nombre d'agent infectés pour chaque tick
V : vecteur contenant le nombre d'agent vaccinés pour chaque tick
"""

function figureEtatSelonTemps(S, I, V)
    f = Figure()
    ax = Axis(f[1, 1]; xlabel="Générations", ylabel="Population")
    stairs!(ax, 1:tick, S, label="Susceptibles", color=:black)
    stairs!(ax, 1:tick, I, label="Infectieux", color=:red)
    stairs!(ax, 1:tick, V, label="Vaccinés", color=:blue)
    axislegend(ax)
    current_figure()
end

"""
    function figureEvent(events, typeevent::String)

Figure permettant de voir quand et où ont eu lieux certains évenements

events : object contenant tous les infromations des évenements d'un certain type
typeevent : Séries de caractères contenant le type d'event observé pour le titre de la figure
"""

function figureEvent(events, typeevent::String)
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:92:48

function figureEvent(events, typeevent::String)
#                                              └ ── premature end of input
```

Si events ne contient aucune valeur, affiche un message d'erreur

```
    if length(events) > 0
        t = [event.time for event in events]
        pos = [(event.x, event.y) for event in events]

        f = Figure()
        ax = Axis(f[1, 1]; aspect=1, backgroundcolor=:grey97)
        hm = scatter!(ax, pos, color=t, colormap=:navia, strokecolor=:black, strokewidth=1,
colorrange=(0, tick), markersize=6)
        Colorbar(f[1, 2], hm, label="Time of $typeevent")
        hidedecorations!(ax)
        current_figure()
    else
        @warn "Il n'y a aucune information enregistrée dans l'event"
    end
end

"""
    function figureBudget(budget, RAT, vaccin)

Figure permettant d'observer le budget et les dépenses à travers le temps

budget : vecteur contenant l'argent restant pour chaque tick
RAT : vecteur contenant les dépenses des tests RAT pour chaque tick
vaccin : vecteur contenant les dépenses des vaccins pour chaque tick
"""

function figureBudget(budget, RAT, vaccin)
    f = Figure()
    ax = Axis(f[1, 1]; xlabel="Générations", ylabel="Dollards")
    stairs!(ax, 1:tick, budget, label="Argents restants", color=:green)
    stairs!(ax, 1:tick, RAT, label="Dépenses RAT", color=:red)
    stairs!(ax, 1:tick, vaccin, label="Dépenses Vaccins", color=:blue)
    axislegend(ax)
    current_figure()
end

"""
    function histogramme(vecteur, titre::String)


Histogramme qui renvoie la fréquence des valeurs contenue dans un vecteur

vecteur : une série de valeur qui sera utilisé dans l'histogrammme
titre : séries de caractères qui contient le titre pour l'histogramme, dépend du vecteur
envoyé
"""

function histogramme(vecteur, titre::String)
    h = Figure()
    hist(h[1, 1], vecteur, color=:grey40, axis=(title="$titre", xlabel="Nombre d'Argents",
ylabel="Fréquences"))
    current_figure()
end
"""
```

```
    function VoisinsMort(mort::Agent, rayon::Integer)

Fonction qui renvoie les voisins non vaccinés d'un agent mort dans un rayon choisi selon la
distance

mort : un agent qui viens de mourrir et autour duquel nous allons trouver les voisins
rayon : la distance autour duquel nous allons chercher pour les agents voisins
"""

function VoisinsMort(mort::Agent, rayon::Integer)
    popVoisins = Agent[]
```

```
ERROR: MethodError: no method matching length(::typeof(Makie.events))
The function `length` exists, but no method is defined for this combination of argument types.

Closest candidates are:
  length(!Matched::Tables.DictRowTable)
   @ Tables ~/.julia/packages/Tables/cRTb7/src/dicts.jl:118
  length(!Matched::Markdown.MD)
   @ Markdown /opt/hostedtoolcache/julia/1.12.5/x64/share/julia/stdlib/v1.12/Markdown/src/
parse/parse.jl:35
  length(!Matched::LaTeXStrings.LaTeXString)
   @ LaTeXStrings ~/.julia/packages/LaTeXStrings/6NrIG/src/LaTeXStrings.jl:112
  ...
```

Boucle qui passe au travers de tous les agents de la population

```
    for agent in population
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:1:28
    for agent in population
#                          └ ── premature end of input
```

Si l'agent se retrouve dans un rayon choisi de l'agent mort et qu'il n'est pas vacciné, on l'ajoute au vecteur

```
        if abs(agent.x - mort.x) < rayon && abs(agent.y - mort.y) < rayon && !
isvaccinated(agent)
            push!(popVoisins, agent)
        end
    end
    return popVoisins
end

"""
    function RATPopulation(pop::Population)

Fonction qui test une population d'agent non testé et renvoie les agents qui ont testé positif
aux test
La fonction n'effectue les tests que s'il reste assez d'argent dans le budget puis note les
dépenses

pop : Population d'agent qui avoisine un agent qui viens de mourir et auquels nous allons
effecter des RAT
```

```
"""

function RATPopulation(pop::Population)
    global budget
    global depenseRAT
    popPositif = Agent[]
```

```
ERROR: UndefVarError: `agent` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

Boucle qui passe au travers de la population reçu en argument, mais seulement ceux qui n'ont pas été testés

```
    for agent in nottested(pop)
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:1:32
    for agent in nottested(pop)
#                              └ ── premature end of input
```

S'assure qu'on a assez de budget pour faire un RAT sur l'agent, sinon on arrete les tests

```
        if budget > coutRAT
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:1:28
        if budget > coutRAT
#                           └ ── premature end of input
```

Enlève le cout du test

```
            budget -= coutRAT
```

```
ERROR: UndefVarError: `budget` not defined in `Main.var"##277"`
Suggestion: add an appropriate import or assignment. This global was declared but not
assigned.
```

Ajout le cout aux dépenses

```
            depenseRAT += coutRAT
```

```
ERROR: UndefVarError: `depenseRAT` not defined in `Main.var"##277"`
Suggestion: add an appropriate import or assignment. This global was declared but not
assigned.
```

Enregistre que cet agent a été testé pour ce tick, pour ne pas le retesté si plusieurs morts

```
            agent.tested = true
```

```
ERROR: UndefVarError: `agent` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

Enregistre l'événement de vaccination

```
            push!(eventsRAT, RATEvent(tick, agent.id, agent.x, agent.y))
```

```
ERROR: UndefVarError: `eventsRAT` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

Renvoie un vrai positif pour une probabilité de "efficaciteRAT" si l'agent est infecté

```
            if efficaciteRAT > rand() && isinfectious(agent)
                push!(popPositif, agent)
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:2:41
            if efficaciteRAT > rand() && isinfectious(agent)
                push!(popPositif, agent)
#                                     └ ── Expected `end`
```

Renvoie un faux positif pour une probabilité de 1 - "efficaciteRAT" si l'agent est sain

```
            elseif efficaciteRAT < rand() && ishealthy(agent)
                push!(popPositif, agent)
            end
        else
            break
        end
    end
    return popPositif
end

"""
    function VaccinPopulation(popVaccin::Population)

Fonction qui vaccine la population totale selon la population reçu en argument
```

```
La fonction n'effectue les vaccinations que s'il reste assez d'argent dans le budget puis note
les dépenses
Les agents qui se font vacciner ont leur temps de latence de vaccination ajuster 2, le temps
d'attente avant que le vaccin fasse effet

pop : Population d'agent qui ont testé positif aux test RAT
"""

function VaccinPopulation(popVaccin::Population)
    global budget
    global depenseVaccin
    for agent in popVaccin
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:1:13
            elseif efficaciteRAT < rand() && ishealthy(agent)
#           └────┘ ── invalid identifier
```

S'assure qu'on a assez d'argent pour vacciner l'agent, sinon arrête les vaccinations

```
        if budget > coutVaccin
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:1:31
        if budget > coutVaccin
#                             └ ── premature end of input
```

Enlève le cout du test

```
            budget -= coutVaccin
```

```
ERROR: UndefVarError: `budget` not defined in `Main.var"##277"`
Suggestion: add an appropriate import or assignment. This global was declared but not
assigned.
```

Ajoute le coût aux dépenses

```
            depenseVaccin += coutVaccin
```

```
ERROR: UndefVarError: `depenseVaccin` not defined in `Main.var"##277"`
Suggestion: add an appropriate import or assignment. This global was declared but not
assigned.
```

Enregistre l'événement de vaccination

```
            push!(eventsVac, VaccinationEvent(tick, agent.id, agent.x, agent.y))
```

```
ERROR: UndefVarError: `eventsVac` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

Ajuste le temps de lattence avant que le vaccin fasse effet

```
            agent.vaccinationclock = 2
        else
            break
        end
    end
end
```

```
ERROR: UndefVarError: `agent` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

Fonction qui note les voisins d'une population d'agents morts, les testent et vaccinent ceux qui sont positifs dans un rayon donné

```
"""
    function Vaccination(popMort::Population, rayon::Int)

Fonction qui effectue tous les étapes de la vaccintaion, soit trouver les agents voisins d'une
population d'agent mort,
les tests, puis vaccinent  ceux qui ont testé positif

popMort : Population d'agent mort aux dernier tick
rayon : Distance dans laquel nous allons chercher pour les voinsins des agents morts
"""
function Vaccination(popMort::Population, rayon::Int)
    for mort in popMort
        VaccinPopulation(RATPopulation(VoisinsMort(mort, rayon)))
    end
end
```

```
Main.var"##277".Vaccination
```

Création de différent type afin de stocker les événements d'infections, de vaccinations et RAT

```
struct InfectionEvent
    time::Int64
    from::UUIDs.UUID
    to::UUIDs.UUID
    x::Int64
    y::Int64
end

struct VaccinationEvent
    time::Int64
    who::UUIDs.UUID
```

```
    x::Int64
    y::Int64
end

struct RATEvent
    time::Int64
    who::UUIDs.UUID
    x::Int64
    y::Int64
end
```

Simulations

Nous allons simuler le comportement d'une épidémie, qui se transmet par contact direct, et qui entraîne la mort après un intervale de temps fixe.

Nous allons maintenant créer un paysage de départ:

```
L = Landscape(xmin=-50, xmax=50, ymin=-50, ymax=50)
```

```
Main.var"##277".Landscape(-50, 50, -50, 50)
```

Création de nouvelles fonctions

On va commencer par générer une fonction pour créer des agents au hasard. Il existe une fonction pour faire ceci dans *Julia*: `rand`. Pour que notre code soit facile a comprendre, nous allons donc ajouter une méthode à cette fonction:

```
Random.rand(::Type{Agent}, L::Landscape) = Agent(x=rand(L.xmin:L.xmax), y=rand(L.ymin:L.ymax))
Random.rand(::Type{Agent}, L::Landscape, n::Int64) = [rand(Agent, L) for _ in 1:n]
```

Nous pouvons maintenant définir des fonctions qui vont nous permettre de nous simplifier la rédaction du code.

Vérifie si un individu est infectieux (malade et contagieux)

```
isinfectious(agent::Agent) = agent.infectious
```

```
isinfectious (generic function with 1 method)
```

Et on peut donc vérifier si un agent est sain:

```
ishealthy(agent::Agent) = !isinfectious(agent) && !isvaccinated(agent)
```

```
ishealthy (generic function with 1 method)
```

Vérifie si un agent est vaccinated

```
isvaccinated(agent::Agent) = agent.vaccinated
```

```
isvaccinated (generic function with 1 method)
```

Vérifie si un agent est testé

```
isnottested(agent::Agent) = !agent.tested
```

```
isnottested (generic function with 1 method)
```

On peut maintenant définir une fonction pour prendre uniquement les agents qui sont infectieux dans une population. Pour que ce soit clair, nous allons créer un *alias*, `Population`, qui voudra dire `Vector{Agent}`:

Retourne les individus d'un certains états d'une population

```
infectious(pop::Population) = filter(isinfectious, pop)
healthy(pop::Population) = filter(ishealthy, pop)
vaccinated(pop::Population) = filter(isvaccinated, pop)
nottested(pop::Population) = filter(isnottested, pop)
```

```
nottested (generic function with 1 method)
```

Nous allons enfin écrire une fonction pour trouver l'ensemble des agents d'une population qui sont dans la même cellule qu'un agent:

```
"""
Retourne les individus présents dans la même cellule qu'un agent.

Paramètres :
- target : agent cible
- pop : population

Retour :
- Liste des agents au même endroit

Biologiquement, cela représente les contacts directs
nécessaires à la transmission.
"""

incell(target::Agent, pop::Population) = filter(ag -> (ag.x, ag.y) == (target.x, target.y),
pop)
```

```
incell (generic function with 1 method)
```

Paramètres initiaux

On en profite pour simplifier l'affichage de cette population:

```
Base.show(io::IO, ::MIME"text/plain", p::Population) = print(io, "Une population avec
$(length(p)) agents")

nbSim = 50
mortFinale = zeros(Int, nbSim)
coutFinale = zeros(Int, nbSim)

taillepop = 3750
population = Population[]
tick = 0
maxlength = 2000
budget = 21_000
coutRAT = 4
coutVaccin = 17
depenseRAT = 0
depenseVaccin = 0
efficaciteRAT = 0.95
```

```
0.95
```

Pour étudier les résultats de la simulation, nous allons stocker la taille de populations à chaque pas de temps:

```
S = zeros(Int64, maxlength)
I = zeros(Int64, maxlength)
V = zeros(Int64, maxlength)
```

```
2000-element Vector{Int64}:
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
```

Pour étudier comment l'argent est dépensé

```julia
budgetVecteur = zeros(Int64, maxlength)
depenseRATVecteur = zeros(Int64, maxlength)
depenseVaccinVecteur = zeros(Int64, maxlength)
```

```
2000-element Vector{Int64}:
 0
 0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

Mais nous allons aussi stocker tous les évènements d'infections, vaccinations et RAT qui ont lieu pendant la simulation:

```
eventsInf = InfectionEvent[]
eventsVac = VaccinationEvent[]
eventsRAT = RATEvent[]
```

```
Main.var"##277".RATEvent[]
```

Boucle afin de faire la simluation plusieurs fois

```
for i in 1:nbSim
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:1:17
for i in 1:nbSim
#               └ ── premature end of input
```

Réinitialse les valeurs de chacun de ces paramètres avant chaque simulation

```
    global tick = 0
    global budget = 21_000
    global depenseRAT = 0
    global depenseVaccin = 0
    global eventsInf = InfectionEvent[]
    global eventsVac = VaccinationEvent[]
    global eventsRAT = RATEvent[]
    global S = zeros(Int64, maxlength)
    global I = zeros(Int64, maxlength)
    global V = zeros(Int64, maxlength)
    global budgetVecteur = zeros(Int64, maxlength)
    global depenseRATVecteur = zeros(Int64, maxlength)
    global depenseVaccinVecteur = zeros(Int64, maxlength)
```

```
2000-element Vector{Int64}:
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
 0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```

Et on génère notre population aléatoire initiale:

```
    global population = Population(L, taillepop)
```

```
Une population avec 3750 agents
```

Pour commencer la simulation, il faut identifier un cas index, que nous allons choisir au hasard dans la population:

Sélection aléatoire d'un individu initialement infecté (cas index)

```
    rand(population).infectious = true
```

```
true
```

Simulation

```
    """
    Boucle principale de la simulation.

    À chaque itération :
    - les individus se déplacent
    - les infections ont lieu
    - les individus infectés progressent vers la mort
    - les morts sont retirés

    La simulation s'arrête lorsqu'il n'y a plus d'infecté
    ou lorsque la durée maximale est atteinte.
    """

    while (length(infectious(population)) != 0) & (tick < maxlength)

        # On spécifie que nous utilisons les variables définies plus haut
```

```
        global tick, population, budget, depenseRAT, depenseVaccin, eventsInf, eventsVac,
eventsRAT

        tick += 1
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:20:18

        tick += 1
#               └ ── Expected `end`
```

Déplacement des individus : Chaque individu se déplace aléatoirement dans l'espace

```
        # Movement

        for agent in population
            move!(agent, L; torus=false)
        end
```

Transmission de la maladie : Les individus infectieux peuvent contaminer les individus sains présents dans la même cellule avec une probabilité de 0.4

```
        # Infection

        for agent in Random.shuffle(infectious(population))
            neighbors = healthy(incell(agent, population))
            for neighbor in neighbors
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:5:38
            neighbors = healthy(incell(agent, population))
            for neighbor in neighbors
#                                   └ ── premature end of input
```

Infecté par une probabilité de 0,4 ET s'il n'est pas vacciné

```
                if rand() <= 0.4 && !isvaccinated(neighbor)
                    neighbor.infectious = true
                    push!(eventsInf, InfectionEvent(tick, agent.id, neighbor.id, agent.x,
agent.y))
                end
            end
        end
```

```
ERROR: UndefVarError: `neighbor` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

Progression de la maladie : Le temps de vie avant la mort diminue pour chaque individu infecté

```
        # Change in survival

        for agent in infectious(population)
            agent.infectionclock -= 1
        end

        # Note tous les agents qui sont morts ce tick

        popMort = filter(x -> x.infectionclock == 0, population)

        # Suppression des individus morts : Les individus dont le temps est écoulé sont
retirés de la population

        population = filter(x -> x.infectionclock > 0, population)

        # Change l'état de l'agent à vacciner et non-infecté lorsque la vaccin commence à
faire effet

        for agent in population
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:17:32

        for agent in population
#                               └ ── premature end of input
```

Réinitialse l'état de l'agent non-testé s'il a été testé au tick précédent

```
            agent.tested = false
```

```
ERROR: UndefVarError: `agent` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

Si le temps de latence du vaccin est supérieur à 0, on le décrémente

```
            if agent.vaccinationclock > 0
                agent.vaccinationclock -= 1
```

```
ERROR: ParseError:
# Error @ /home/runner/work/Devoir_3/Devoir_3/travail.md:2:44
            if agent.vaccinationclock > 0
                agent.vaccinationclock -= 1
#                                          └ ── Expected `end`
```

Si le temps de latence du vaccin atteint 0, la personne devient vacciné et perd son état d'infecté si elle l'était

```
                if agent.vaccinationclock == 0
                    agent.vaccinated = true
```

```
                    agent.infectious = false
                end
            end
        end
```

```
ERROR: UndefVarError: `agent` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

S'il reste du budget, fait une vaccination et RAT des agents lorsqu'il y a un mort

```
        if budget > coutRAT && length(popMort) > 0
            Vaccination(popMort, 21)
        end
```

Enregistrement des données : Stock le nombre d'individus sains et infectés à chaque instant

```
        S[tick] = length(healthy(population))
        I[tick] = length(infectious(population))
        V[tick] = length(vaccinated(population))
```

```
ERROR: BoundsError: attempt to access 2000-element Vector{Int64} at index [0]
```

Enregistrement des données : Stock les dépenses de chaque instant

```
        budgetVecteur[tick] = budget
        depenseRATVecteur[tick] = depenseRAT
        depenseVaccinVecteur[tick] = depenseVaccin


    end
```

```
ERROR: BoundsError: attempt to access 2000-element Vector{Int64} at index [0]
```

Coupe la longueur des informations au dernier tick où de l'infromation à été enregistré

```
    S = S[1:tick]
    I = I[1:tick]
    V = V[1:tick]
    budgetVecteur = budgetVecteur[1:tick]
    depenseRATVecteur = depenseRATVecteur[1:tick]
    depenseVaccinVecteur = depenseVaccinVecteur[1:tick]
```

```
Int64[]
```

Enregistre le nombre de mort ainsi que le coût total à la fin de la simulation

```
    mortFinale[i] = taillepop - length(S) - length(I) - length(V)
    coutFinale[i] = budgetVecteur[1] - budget

end
```

```
ERROR: UndefVarError: `i` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

Analyse des résultats

Figure 1 : Représentation graphique des dépenses et du budget d'une seule simluation à travers le temps

```
figureBudget(budgetVecteur, depenseRATVecteur, depenseVaccinVecteur)
```

```
ERROR: UndefVarError: `figureBudget` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

Figure 2 : Représentation graphique des différents états des agents à travers le temps

```
figureEtatSelonTemps(S, I, V)
```

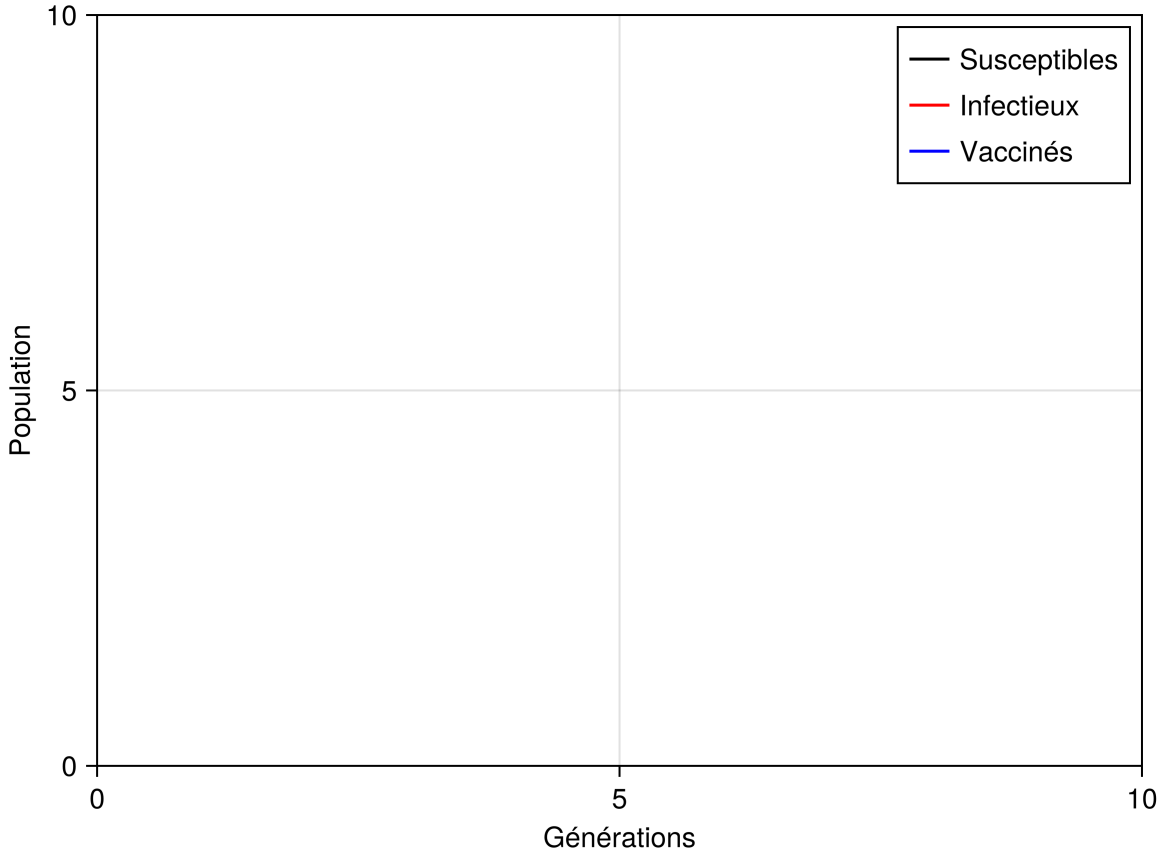

Figure 3 : Histogramme du nombre d'agents morts à la fin de 250 simulations

```
histogramme(mortFinale, "Morts")
```

```
ERROR: UndefVarError: `histogramme` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

Figure 4 : Histogramme du coût total à la fin de 250 simulations

```
histogramme(coutFinale, "Dépenses")
```

```
ERROR: UndefVarError: `histogramme` not defined in `Main.var"##277"`
Suggestion: check for spelling errors or missing imports.
```

Présentation des résultats

Discussion

On peut aussi citer des références dans le document `references.bib`, qui doit être au format BibTeX. Les références peuvent être citées dans le texte avec `@` suivi de la clé de citation. Par exemple: [9] – la bibliographie sera ajoutée automatiquement à la fin du document.

Le format de la bibliographie est American Physics Society, et les références seront correctement présentées dans ce format. Vous ne devez/pouvez pas éditer la bibliographie à la main

Bibliographie

[1] C. Fraser, S. Riley, R. M. Anderson, et N. M. Ferguson, Factors that make an infectious disease outbreak controllable, Proceedings of the National Academy of Sciences **101**, 6146 (2004).

[2] K. E. Jones, N. G. Patel, M. A. Levy, A. Storeygard, D. Balk, J. L. Gittleman, et P. Daszak, Global trends in emerging infectious diseases, Nature **451**, 990 (2008).

[3] H. W. Hethcote, The Mathematics of Infectious Diseases, SIAM Review **42**, 599 (2000).

[4] X. He, E. H. Y. Lau, P. Wu, X. Deng, J. Wang, X. Hao, et others, Temporal dynamics in viral shedding and transmissibility of COVID-19, Nature Medicine **26**, 672 (2020).

[5] K. Chalkidou et al., Priority-setting for achieving universal health coverage, Bulletin of the World Health Organization **94**, 462 (2016).

[6] F. P. Polack, S. J. Thomas, N. Kitchin, et others, Safety and Efficacy of the BNT162b2 mRNA Covid-19 Vaccine, New England Journal of Medicine **383**, 2603 (2020).

[7] A. Scohy, A. Anantharajah, M. Bodéus, B. Kabamba-Mukadi, A. Verroken, et H. Rodriguez-Villalobos, Low performance of rapid antigen detection test as frontline testing for COVID-19 diagnosis, Journal of Clinical Virology **129**, 104455 (2020).

[8] C. M. Peak, L. M. Childs, Y. H. Grad, et C. O. Buckee, Comparing nonpharmaceutical interventions for containing emerging epidemics, Proceedings of the National Academy of Sciences **114**, 4023 (2017).

[9] G. B. Ermentrout et L. Edelstein-Keshet, Cellular Automata Approaches to Biological Modeling, Journal of Theoretical Biology **160**, 97 (1993).