

**DOCUMENTATION INTERNE**

**Projet :**

**Modelling and simulating**

**human behavioural factors**

**influencing the spread of Covid-19**

Rémi HUYGHE

Moustafa MOHAMED

Maminiaina RAZANAMARIMANDIMBY

**UNIVERSITÉ GRENOBLE ALPES**

**DOCUMENTATION INTERNE**

**Modelling and simulating human behavioural factors**

**influencing the spread of Covid-19**

| Référence du document | D6 |
| --- | --- |
| Version du document : | 1 |
| Date du document : | 26 Mai 2021 |
| Auteurs : | Rémi HUYGHE  Moustafa MOHAMED  Maminiaina RAZANAMARIMANDIMBY |

| Validé par : |  |
| --- | --- |
| Validé le : |  |
| Soumis le : |  |
| Type de diffusion : | Document électronique partagé  (Google Drive) |
| Confidentialité : | Chef de projet  Membres de l’équipe |

| **Mots clés : Documentation interne** |
| --- |

**Sommaire**

[**I. Préambule**](#_o5tdq8scacn9) **6**

[I.1. Objectifs et méthodes](#_pgcqc0sxz9zt) 6

[I.2. Documents de référence](#_pmqhvspk4g41) 6

[**II. Guide de lecture**](#_99geq51b0efd) **6**

[**III. Concepts de base**](#_pf4twoduxdf) **7**

[III.1. Les agents](#_axbpmw34l705) 7

[III.2. Les fonctions](#_cal4l7xgamno) 7

[III.3. Autres précisions](#_ktifp9ogglja) 7

[**IV. Globals**](#_6f3hqi7b2pjt) **8**

[IV.1. Objectif](#_b0eq4lf7hcyn) 8

[IV.2. Relations d’utilisation avec d’autres modules](#_18i9ub24cabo) 8

[IV.3. Les variables définies](#_2q6w3tvib1mz) 8

[IV.4. Procédures et variables externes](#_n99bv5eiwx5c) 8

[**V. Breed**](#_sczf6c64b421) **9**

[V.1. Objectif](#_itfivk7l29rk) 9

[V.2. Relations d’utilisation avec d’autres modules](#_6pyd2u8m7yhi) 9

[V.3. Les variables définies](#_5w9us390ne20) 9

[V.4. Procédures et variables externes](#_cwjf8n94mvjn) 9

[**VI. Setup**](#_3kaqvcmukrcy) **10**

[VI.1. Objectif](#_wnrq75c3l58x) 10

[VI.2. Relations d’utilisation avec d’autres modules](#_itcff0u8mchs) 10

[VI.3. Les variables définies](#_50edl9iuushf) 10

[VI.4. Procédures et variables externes](#_fk6ew4gn8f9l) 10

[**VII. Go**](#_jfikjxxhpkhj) **11**

[VII.1. Objectif](#_vmu6y62zmf1a) 11

[VII.2. Relations d’utilisation avec d’autres modules](#_drbmhx21dyg) 11

[VII.3. Les variables définies](#_a8npct93vkie) 11

[VII.4. Procédures et variables externes](#_ebuwd4dwgrfg) 11

[**VIII. Setup-globals**](#_40usah361ued) **12**

[VIII.1. Objectif](#_fliyngxh06) 12

[VIII.2. Relations d’utilisation avec d’autres modules](#_dak3k2dhxrsg) 12

[VIII.3. Les variables définies](#_6hc7538jsybj) 12

[VIII.4. Procédures et variables externes](#_vsrhkx19jin0) 12

[**IX. Setup-population**](#_xpft7hh6k6du) **13**

[IX.1. Objectif](#_5ezdzqj29n9e) 13

[IX.2. Relations d’utilisation avec d’autres modules](#_q44j6yhtk5sy) 13

[IX.3. Les variables définies](#_rbbgaie0f8zw) 13

[IX.4. Procédures et variables externes](#_5gj6npm4ov4s) 14

[**X. Avoid-walls**](#_g85j47xnwux8) **14**

[X.1. Objectif](#_omha1k61d786) 14

[X.2. Relations d’utilisation avec d’autres modules](#_26epbhjs4jdy) 14

[X.3. Les variables définies](#_ogl1qh8dbmoe) 14

[X.4. Procédures et variables externes](#_tnjn1dw12b2b) 14

[**XI. Set-infected-initialisation**](#_1235jbaelmx3) **15**

[XI.1. Objectif](#_eebf2rwbm8mp) 15

[XI.2. Relations d’utilisation avec d’autres modules](#_60cv4sz9ap54) 15

[XI.3. Les variables définies](#_3ydypdpd2ye9) 15

[XI.4. Procédures et variables externes](#_lv7nynkk7sju) 15

[**XII. Show-epidemic-state**](#_woozehfzymbj) **16**

[XII.1. Objectif](#_cuyvp8wxmvn1) 16

[XII.2. Relations d’utilisation avec d’autres modules](#_2ot02if6re2k) 16

[XII.3. Les variables définies](#_u3jt0efnbtey) 16

[XII.4. Procédures et variables externes](#_fy9yaqqxh2nj) 16

[**XIII. Avoid-infected**](#_jeodpf3nm5z3) **17**

[XIII.1. Objectif](#_lilfea61i5rd) 17

[XIII.2. Relations d’utilisation avec d’autres modules](#_b9xnmh6t1fc) 17

[XIII.3. Les variables définies](#_6csridgxr80l) 17

[XIII.4. Procédures et variables externes](#_nsort0w95c8q) 18

[**XIV. Quarantine-infected**](#_624rw92j8ws4) **19**

[XIV.1. Objectif](#_cdu2ndqgjrom) 19

[XIV.2. Relations d’utilisation avec d’autres modules](#_593qpm6sd8rs) 19

[XIV.3. Les variables définies](#_79kq4qvmuc0) 19

[XIV.4. Procédures et variables externes](#_3stp14d8np8g) 19

[**XV. Recover**](#_ggmm3va2odhw) **19**

[XV.1. Objectif](#_vh7lk4z1zs6a) 20

[XV.2. Relations d’utilisation avec d’autres modules](#_dbf38fs895) 20

[XV.3. Les variables définies](#_3heotc5zfqpx) 20

[XV.4. Procédures et variables externes](#_g38k9rd0qsg4) 20

[**XVI. Interventions-off**](#_mpg269umpee8) **21**

[XVI.1. Objectif](#_r8w32bqmm15) 21

[XVI.2. Relations d’utilisation avec d’autres modules](#_ofg1mx8colnh) 21

[XVI.3. Les variables définies](#_y0h4xtnry6x) 21

[XVI.4. Procédures et variables externes](#_uewxkwd0yfu) 21

[**XVII. Report-rec, Report-inf, Report-sus**](#_31gdw6270kpp) **21**

[XVII.1. Objectif](#_3jmed7hi2ic1) 21

[XVII.2. Relations d’utilisation avec d’autres modules](#_dub03gy9daz6) 21

[XVII.3. Les variables définies](#_i4mjhrlbcvdg) 21

[XVII.4. Procédures et variables externes](#_j3jkiyoyg6ft) 21

[**XVIII. Glossaire**](#_8bnt78oq7am1) **22**

[**XX. Références**](#_ojsp72o0opkf) **24**

# I. Préambule

Ce document est la trace écrite de la programmation de notre simulateur. Il servira de document de maintenance du code de notre programme, à la fois un repère explicitant les différentes composantes du programme, mais aussi une balise pour une reprise potentielle dans le futur.

Ce document a donc pour but de faciliter la maintenance du code écrit en langage de programmation interne à Netlogo.

## I.1. Objectifs et méthodes

C’est un modèle conçu pour être facilement adaptable, les différents paramètres des agents sont modifiables, l’ajout de critères est aisé.

## I.2. Documents de référence

Toute la méthodologie ayant menée à la conception et à la définition des variables initiales de ce document se retrouve dans le Rapport.

Toute la partie pratique et nécessaire à l’installation du logiciel est disponible dans le manuel d’installation.

Toute la partie qui explique l’interface et l’agencement des différents éléments de l’interface est disponible dans le manuel d’utilisation.

# II. Guide de lecture

Pour un utilisateur novice étranger au logiciel Netlogo, un rapide passage par les concepts de base (partie III) est fortement conseillé pour avoir une meilleure compréhension. On y retrouve la définition des termes génériques, spécifiques au logiciel Netlogo et qui sont utilisés tout au long de notre code.

Pour le reste, chaque module est une partie distincte, donc pour une lecture spécifique de chaque module, merci de vous diriger directement à la partie IV.

# III. Concepts de base

Pour une meilleure compréhension, nous reprenons ici quelques termes génériques du logiciel Netlogo.

## III.1. Les agents

Pour rappel, Netlogo est un simulateur de comportements que peuvent avoir des agents. Ici, la notion d’agents renvoie à des “*êtres* capables d’exécuter des instructions. Chaque agent peut avoir sa propre activité, mais tous agissent simultanément.” (Romain Jean, 2009)

Netlogo connaît 4 types d’agents, mais nous nous sommes intéressés aux turtles, ayant comme spécificité la capacité de se déplacer dans un espace défini, le world.

## III.2. Les fonctions

Dans le logiciel Netlogo, les fonctions du modèle sont appelées des **commandes** et de **reporters**. Ces deux types de fonctions disent aux agents - les turtles - ce qu’ils doivent faire (les *commandes*) et communiquent des valeurs calculées (les *reporters*).

Comme dans tout langage de programmation, Netlogo dispose de commandes et de reporters prédéfinies - ce sont les **primitives[[1]](#footnote-0)**.

## III.3. Autres précisions

Notre modèle est articulé en un seul module abstrait durable, afin qu'il puisse être utilisé par différents chercheurs. Dans ce module, chaque utilisateur peut définir les paramètres à sa guise afin de tester essentiellement la vitesse à laquelle le virus se propage. Comme valeurs initiales, le modèle est instancié avec les données récoltées par notre questionnaire, notamment sur les variables descriptives de notre population.

La durabilité de notre code fait que ces paramètres sont modulables et modifiables à la guise de tout utilisateur qui reprend notre programme. Il pourra, selon ce qu’il voudra tester comme scénario, modifier les différents paramètres.

Ici nous allons donc expliciter chaque fonctionnalité ou paramètre que l’utilisateur pourra moduler, avec toute la partie interne au code relative à chacune de ces fonctionnalités.

# IV. Globals

## IV.1. Objectif

Ici, on définit toutes les variables et les constantes globales qui seront utilisées partout dans le programme. Ce mot clé est forcément situé au début du programme pour des raisons de lecture de code et de cohérence. Les variables et constantes déclarées ici sont dites “globales” parce qu’elles sont accessibles par tous les agents, à n’importe quel endroit du programme.

## IV.2. Relations d’utilisation avec d’autres modules

Comme on définit ici les variables globales, tous les autres modules sont donc concernés et liés à celui-ci.

## IV.3. Les variables définies

globals définit les variables globales utilisées dans les différentes fonctions du code qui sont les suivantes :

probability-transmission

Probabilité de transmission de virus d’un agent à un autre.

transmission-distance  
Distance de transmission de virus d’un agent à un autre.

%respect-distanciation  
Pourcentage de la population qui respecte la distanciation sociale.

infected-avoidance-distance  
Distance appliquée par les agents sains envers les agents malades.

walking-angle  
Angle de déplacement des agents dans le world

speed  
Vitesse de déplacement des agents dans le world

transparency  
Niveau de transparence des couleurs des agents.

wall  
Mur virtuel qui aide à garder les agents en mouvement dans sur la zone spécifiée du world

## IV.4. Procédures et variables externes

Comme précisé dans le point IV.1., globals définit toutes les variables et constantes utilisées dans tout le programme, donc tout ce qui est déclaré ici est visible et utilisable par tous les autres modules du programme.

# V. Breed

## V.1. Objectif

breed une variable tortue et lien préprogrammée. Elle contient l'ensemble d'agents formé de toutes les tortues ou de tous les liens de même race que cette tortue ou ce lien. Elle permet, entre autres, de créer de nouvelles races d’agents.

## V.2. Relations d’utilisation avec d’autres modules

Cette variable fonctionne comme globals. Il définit une nouvelle race d’agents, qui sera utilisée dans la suite du programme.

## V.3. Les variables définies

breed définit une nouvelle race d’agents dans notre programme : les citizens.

Grâce à breed, nous avons une nouvelle variable citizens-own, correspondant aux différents attributs des citizens :

epidemic-state  
État épidémique de chaque citizen. Cet attribut a comme modalité : “Susceptible” et “Infected”

infection-date  
Date (en nombre de jours, de ticks) de l’infection de chaque citizen.

nb-other-infected  
Chaque citizen connaît, grâce à cet attribut, le nombre de citizen déjà malades.

nb-contacts  
Nombre de contacts que chaque citizen a avec les autres citizens.

respect-rules?  
Booléen définissant si le citizen respecte les règles gouvernementales.

quarantined?  
Booléen définissant si le citizen est en quarantaine.

## V.4. Procédures et variables externes

La définition de citizens comme nouvelle race relie cette procédure à la variable externe citizens-own.

# VI. Setup

## VI.1. Objectif

setup est un reporter associé au bouton du même nom, qui initialise l’interface avec les variables définies en dur. Attention, ce reporter doit être exécuté avant de lancer l’exécution du reporter go, et ce à chaque fois.

## VI.2. Relations d’utilisation avec d’autres modules

Ce module fonctionne en lien avec les modules setup-globals et setup-population. Il exécute également les primitives clear-all et reset-ticks.

Comme précisé précédemment, ce module doit être exécuté avant le module go.

## VI.3. Les variables définies

setup ne définit pas de variables en interne. Il ne fait qu’appeler les modules setup-globals et setup-population qu’il exécute, dès que l’utilisateur appuie sur le bouton setup.

## VI.4. Procédures et variables externes

Le reporter setup appelle les procédures externes setup-globals et setup-population.

# VII. Go

## VII.1. Objectif

go est un reporter associé au bouton start, qui lance la simulation avec les données définies suite à l’exécution du reporter setup. Lancer la simulation revient, ici, à lancer les mouvements des agents sur l’écran. Il reste activé tant que la simulation ne se termine pas ou que l’utilisateur clique une nouvelle fois dessus, cela équivaut à un mode automatique plutôt que jour par jour.

## VII.2. Relations d’utilisation avec d’autres modules

Ce module fonctionne en lien avec la commande avoid-infected définie plus bas. Il contient également les primitives tick et stop et les primitives conditionnelles ask et if.

## VII.3. Les variables définies

go ne définit pas de variables en interne. Il lance le mouvement des citizens avec les variables définies par l’exécution de setup. Cette exécution contient trois conditions :

if inf = population-size [stop]

Si le nombre total d’infectés est égal à la taille de la population définie, on arrête la simulation avec la primitive stop.

ask citizens with [epidemic-state = “Susceptible”] [get-virus]  
Pour les citizens ayant un état épidémique égal à “Susceptible”, exécuter la commande get-virus.

ask citizens with [epidemic-state = “Infected”] [recover]  
Pour les citizens ayant un état épidémique égal à “Infected”, exécuter la commande recover.

## VII.4. Procédures et variables externes

Le reporter go appelle les procédures externes recover et get-virus.

Il utilise, pour chaque citizen qu’il exécute, son attribut epidemic-state.

Il utilise également la constante externe inf, qui reprend le nombre total d’infectés de la population.

# VIII. Setup-globals

## VIII.1. Objectif

setup-globals est un module qui instancie les variables globales définies par globals. Il permet essentiellement de donner la valeur initiale à tous les paramètres définis par globals.

## VIII.2. Relations d’utilisation avec d’autres modules

Ce module reprend et redéfinit toutes les variables définies dans globals. Et inversement, setup-globals est utilisé par setup.

## VIII.3. Les variables définies

setup-globals ne définit pas de nouvelles variables en interne. Elle attribue des valeurs aux variables globales définies par globals. Ici, on a pris le choix de définir les variables initiales aux valeurs recueillies dans notre questionnaire :

set population-size 153

Nous avons 153 répondants à notre questionnaire.

set nb-infected-initialisation 10  
On compte 10 citizens infectés, à l’initialisation.

set distanciation-distance 1.5  
La distance de distanciation sociale au départ est fixée à 1.5 mètre.

set %respect-distanciation 90  
On considère que 90% des répondants appliquent la distanciation sociale.

set infected-avoidance-distance 2  
On considère que les citizens évitent les personnes malades à une distance de 2 mètres.

set walking-angle 5  
On considère que les citizens se déplacent avec un angle de 5.

set speed 0.5  
La vitesse de déplacement est instanciée à 0.5.

set wall 0  
La taille des murs autour de l’écran est de 0.

set transparency 145  
Le niveau de transparence de la couleur des citizens est à 145.

## VIII.4. Procédures et variables externes

setup-globals redéfinit les variables instanciées par globals. Chaque définition de variable est préfixée par la primitive set.

# IX. Setup-population

## IX.1. Objectif

setup-population est la commande qui permet de créer les citizens grâce à la primitive create. Elle permet aussi de définir les spécificités de la population, spécificités graphiques et les différents attributs de la taille de la population.

## IX.2. Relations d’utilisation avec d’autres modules

setup-population utilise les citizens définis par breed plus haut.

En lien avec globals, elle ajoute des attributs à la variable globale population-size et instancie les valeurs initiales de epidemic-state, respect-rules? et quarantined?

Elle fait aussi appel à la commande set-infected-initialisation

## IX.3. Les variables définies

set-population ajoute des attributs à la variable population-size de chaque citizen :

setxy random-xcor random-ycor

Des coordonnées de position spatiale dans l’écran dédié au mouvement des citizens.  
Ces coordonnées sont générées de manière aléatoire, grâce aux primitives random-xcor et random-ycor.

set shape “person”  
Instancie les citizens à la forme prédéfinie de person.

set color green  
Définit la couleur de chaque citizen à la couleur verte. Cette couleur peut se changer à la guise de celui qui veut reprendre le code, avec des valeurs prédéfinies. Pour plus d’informations sur les couleurs possibles, veuillez vous référer au Dictionnaire NetLogo.

set size 1.5  
Définit la taille graphique de chaque citizen à 1.5 avec la variable préprogrammée size.

set epidemic-state "Susceptible"

Instancie l’état épidémique de chaque citizen à la valeur “Susceptible”

set respect-rules? true

Définit le booléen respect-rules? à true. En ce sens, chaque citizen respecte les règles établies, par défaut.

set quarantined? false

Définit le booléen quarantined? à false. En ce sens, aucun citizen n’est en quarantaine à l’initiation du code, par défaut.

## IX.4. Procédures et variables externes

setup-population instancie de nouveaux attributs à la variable population-size.

Elle fait appel à la commande set-infected-initialisation.

Elle utilise aussi les primitives set et setxy.

# X. Avoid-walls

## X.1. Objectif

avoid-walls est une commande qui permet à chaque citizen de ne pas sortir de la zone de mouvement de l’écran.

## X.2. Relations d’utilisation avec d’autres modules

Cette commande est attribuée aux citizens et elle s’applique sur les patchs.

## X.3. Les variables définies

avoid-walls contient deux conditions :

if abs [pxcor] of patch-ahead (wall + 2) = max-pxcor [   
set heading (- heading) ]  
On vérifie si la valeur absolue de l’abscisse du patch situé à 2 unités de distance du mur est égale à la valeur maximale de cette abscisse.  
Si cette valeur est vérifiée, le citizen rebrousse chemin (- heading).

if abs [pycor] of patch-ahead 1 = max-pycor [   
set heading (180 - heading) ]  
On vérifie si la valeur absolue de l’ordonnée du patch situé à 1 unité de distance du citizen est égale à la valeur maximale de cette ordonnée  
Si cette valeur est vérifiée, le citizen tourne de 180° de sa position précédente.

## X.4. Procédures et variables externes

avoid-walls n’utilise pas des variables externes.  
Elle prend en valeur de test les coordonnées de position de chaque citizen, ainsi que celles des patchs et du mur (wall).  
Elle définit en sortie la direction (heading) prise par chaque citizen, si les conditions sont vérifiées

# XI. Set-infected-initialisation

## XI.1. Objectif

set-infected-initialisation est une commande qui instancie le nombre de citizen infectés.

## XI.2. Relations d’utilisation avec d’autres modules

Cette commande fait appel à la constante nb-infected-initialisation en valeur d’entrée, grâce à la primitive ask afin de savoir combien y-a-t-il de citizen infectés à l’initialisation du code.

Par la suite, il est en lien avec les citizens auxquels il modifie les attributs epidemic-state et color.

## XI.3. Les variables définies

set-infected-initialisation attribue aux citizens, du nombre de nb-infected-initialisation, l’état épidémique de "Infected" et la couleur 15.

## XI.4. Procédures et variables externes

Cette commande utilise les primitives ask et set et le reporter n-of avec comme constante en entrée nb-infected-initialisation.

# XII. Show-epidemic-state

## XII.1. Objectif

show-epidemic-state définit, pour chaque état des citizens, la couleur correspondante.

## XII.2. Relations d’utilisation avec d’autres modules

show-epidemic-state est utilisée par les commandes recover et get-virus.

## XII.3. Les variables définies

show-epidemic-state s’articule sous forme de conditions imbriquées :

epidemic-state = "Susceptible" [   
set color lput transparency extract-rgb green ]

Si le citizen est “Susceptible”, sa couleur sera green.

epidemic-state = "Infected" [   
set color lput transparency extract-rgb red ]

Si le citizen est “Infected”, sa couleur sera red.

epidemic-state = "Recovered" [   
set color lput transparency extract-rgb white ]

Si le citizen est “Recovered”, sa couleur sera white.

## XII.4. Procédures et variables externes

Cette commande utilise les primitives ifelse et set.

# XIII. Avoid-infected

## XIII.1. Objectif

avoid-infected est une commande qui permet aux citizens de ne pas se rapprocher de ceux qui sont infectés. Elle modélise l'auto-discipline de chaque citizen à s’isoler eux-mêmes des personnes infectées.

## XIII.2. Relations d’utilisation avec d’autres modules

avoid-infected est utilisée par la commande go.

## XIII.3. Les variables définies

Pour chaque citizen :

ask citizens [

let target min-one-of other citizens with [epidemic-state = "Infected"] in-radius infected-avoidance-distance [distance myself]

On définit une variable locale target qui reprend la distance du citizen en question avec le plus petit parmi les autres citizens qui a un état “Infected” dans un environ radial autour de lui, d’une distance définie par la infected-avoidance-distance.

ifelse epidemic-state != "Infected" and is-agent? target

On vérifie si l’agent devant nous n’est pas “Infected” et si target est un agent.

[ face target

rt 180

avoid-walls

]

On fait pivoter target de 180°.  
On évite de se cogner aux murs avec la fonction avoid-walls.

[

set heading   
 heading  
 + random walking-angle   
 - random walking-angle

avoid-walls

]

On définit la direction (heading) comme étant la somme entre la direction courante et un walking-angle aléatoire.  
On évite de se cogner aux murs avec la fonction avoid-walls.

ifelse epidemic-state != "Infected" [ fd speed ]

Si l’agent devant nous n’est pas “Infected”, on avance à la vitesse speed.

[ ifelse member? "off" models [ quarantine-infected ] [ fd speed / 2 ] ]

Sinon, la vitesse de déplacement est divisée de moitié.

## XIII.4. Procédures et variables externes

avoid-infected utilise les attributs epidemic-state, speed et walking-angle des citizens.

Elle fait appel à la commande avoid-walls.

Elle utilise les primitives ask, distance, ifelse, in-radius, let, min-one-of, myself, other, set et with.

# XIV. Quarantine-infected

## XIV.1. Objectif

L’idée de cette commande est d’isoler les citizens infectés et de les placer en quarantaine pour une durée donnée afin qu'elles ne transmettent pas le virus à d'autres citizens.

## XIV.2. Relations d’utilisation avec d’autres modules

Cette commande est utilisée par avoid-infected.

## XIV.3. Les variables définies

Dans cette commande, nous introduisons la variable discovery-period, qui correspond à la durée de temps entre l’infection au virus et la découverte qu’on est positif (assimilé à l’arrivée des premiers symptômes).

if ticks = discovery-period [

if not quarantined? [

set shape "person"

move-to min-one-of patches with [not any? citizens-here] [pxcor]

set size 1.5

set quarantined? true

]

]

Si le nombre de ticks est égal à la valeur de discovery-period et si le citizen n’est pas en quarantaine. On définit sa forme à “person” et sa taille à 1.5. On déplace le citizen en quarantaine et on définit son attribut quarantined? à true.

## XIV.4. Procédures et variables externes

quarantine-infected utilise la constante discovery-period.

Elle utilise également l’attribut quarantined? des citizens pour savoir s’il faut ou non appliquer la quarantaine. Elle définit la taille (size) et la forme (shape) de chaque citizen concerné et le déplace grâce à la primitive move-to min-one-of.

# XV. Recover

## XV.1. Objectif

Cette commande permet aux citizens malades de guérir.

## XV.2. Relations d’utilisation avec d’autres modules

La commande recover est utilisée par la commande go.

## XV.3. Les variables définies

Dans cette commande, nous introduisons la variable recovery-duration, qui correspond à la durée de guérison du virus :

if ticks = recovery-duration + discovery-period

[

ask citizens with [epidemic-state = "Infected"]

[

set color white

set epidemic-state "Recovered"

]

]

show-epidemic-state

Si le nombre de ticks est égal à la somme de discovery-period et de recovery-duration.  
Pour chaque citizen avec une état Infected.

On modifie la couleur à white et l’état à Recovered.

On appelle la commande show-epidemic-state.

## XV.4. Procédures et variables externes

recover utilise les citizens. Elle utilise les primitives set, ask et if  
Elle fait appel à la commande show-epidemic-state.

Elle utilise deux variables externes recovery-duration et discovery-period.

# XVI. Interventions-off

## XVI.1. Objectif

Cette commande, associée au bouton du même intitulé, permet de remettre les compteurs à zéro.

## XVI.2. Relations d’utilisation avec d’autres modules

Il n’y a pas de modules utilisant intervention-off.

## XVI.3. Les variables définies

Cette commande réinitialise trois paramètres, qu’on définit à travers cette commande :

set isolate-symptomatic false

set remove-recovered false

set recovery-duration 0

isolate-symptomatic est un booléen; à valeur true, il signifie que les citizens symptomatiques sont isolés. Ici, on la met à false.  
remove-recovered est un booléen. À valeur true, il signifie que les personnes ayant guéri de la COVID-19 sont retirées.  
recovery-duration définit la durée de guérison de la maladie. Ici, nous avons mis 0.

## XVI.4. Procédures et variables externes

Aucune procédure ni variable externe.

# XVII. Report-rec, Report-inf, Report-sus

## XVII.1. Objectif

Ce sont des reporters qui permettent d’avoir le nombre de citizens qui selon leur état, qu’ils soient “Recovered”, “Infected” ou “Susceptible”.

## XVII.2. Relations d’utilisation avec d’autres modules

Comme ce sont des reporters, ils s'auto-suffisent à eux-mêmes, donc ils n’ont pas besoin d’être liés à d’autres modules.

## XVII.3. Les variables définies

report count citizens with [epidemic-state = "Recovered"]

report count citizens with [epidemic-state = "Infected"]

report count citizens with [epidemic-state = "Susceptible"]

Avec la primitive count, on obtient le nombre pour chacune des valeurs prises par epidemic-state.

## XVII.4. Procédures et variables externes

On utilise ici la primitive count et l’attribut epidemic-state de citizens.

# XVIII. Glossaire

Les primitives essentielles utilisées dans notre programme :

**ask** Commande qui permet à l’observateur de demander à un seul ou à  
 un groupe d’agents d’exécuter les commandes données.

**breed**  C'est une variable tortue et lien préprogrammée.

Elle contient l'ensemble des agents formés de toutes les tortues ou

de tous les liens de même race que cette tortue ou ce lien.

Elle permet, entre autres, de créer de nouvelles races d’agents.

**clear-all** Réinitialise toutes les variables globales à zéro et appelle les primitives suivantes : reset-ticks, clear-turles, clear-patches, clear-drawing, clear-all-plots et clear-output.

**clear-all-plots** Efface le contenu de tous les traceurs de graphiques du modèle.

**clear-drawing** Efface toutes les lignes et tous les tampons dessinés par les tortues

sur la couche de dessin.

**clear-output** Efface tous les textes affichés dans la zone de sortie du modèle, s'il y

en a une. Sinon, ne fait rien.

**clear-patches** Réinitialise les patches.

**clear-turtles** Supprime toutes les tortues.  
 Cette commande réinitialise aussi le numéro d'identification   
 (who number), de manière à ce que la prochaine tortue créée porte

le numéro 0.

**distance** Retourne la distance qu'il y a entre l'agent appelant et la tortue ou au  
 patch spécifié par *agent*.

**fd** ou **forward** La tortue avance du nombre de pas spécifié dans l'entrée *number*,  
 un pas à la fois. (Si *number* est négatif, la tortue recule.)  
 La commande fd 10 est équivalente à repeat 10 [ jump 1 ] et la   
 commande fd 10.5 est équivalente à repeat 10 [ jump 1 ] jump 0.5.  
 Si la tortue ne peut avancer du nombre de pas demandé parce que   
 la topologie du monde NetLogo ne le permet pas, la tortue fait autant   
 de pas de 1 unité qu'elle peut puis s'arrête.

**heading** Variable tortue préprogrammée qui spécifie la direction (le cap) de la  
 tortue. Sa valeur est toujours un nombre plus grand ou égal à zéro et   
 plus petit que 360. 0 indique le nord, 90 l'est, etc, comme en   
 géographie. Vous pouvez modifier cette variable pour faire tourner   
 (pivoter) la tortue sur elle-même.

**if | ifelse** [if] Cette primitive fait exécuter les commandes placées dans la liste  
 [*commands*] si la *condition* testée est vraie. Le reporter *condition* doit  
 retourner une valeur booléenne true ou false  
   
 [ifelse] La primitive ifelse fait exécuter les commandes listées dans   
 [*commands1*] et ignore les commandes listées dans [*commands2*] si   
 la condition *reporter* retourne true (vrai) et ignore les commandes   
 listées dans [*commands1*] mais fait exécuter les commandes listées   
 dans [*commands2*] si la condition *reporter* retourne false (faux).  
 Le reporter *condition* doit retourner une valeur booléenne true ou   
 false (vrai ou faux).  
 Le reporter peut retourner une valeur différente pour des agents   
 différents, ce qui fait que certains agents exécutent les commandes   
 de la liste [*commands1*] alors que d'autres exécutent celles de la liste   
 [*commands2*].

**in-radius** Retourne un ensemble d'agents formé des agents, appartenant au   
 même ensemble d'agents que l'agent appelant, dont la distance à   
 l'appelant est inférieure ou égale à *number*. L'agent appelant peut   
 aussi en faire partie.  
 La distance jusqu'à un patch ou la distance à partir d'un patch est   
 mesurée à partir du centre du patch.

**let** Création d’une nouvelle variable locale dont le nom est spécifié par  
 l'entrée *variable* et lui assigne la valeur spécifiée dans l'entrée *value*.  
 Une variable locale n'existe (n'est visible) que dans le bloc de   
 commandes délimité par des crochets dans lequel elle a été définie.

**min-one-of** Retourne l'agent de l'ensemble d'agent *agentset* qui a la plus petite   
 valeur pour le *reporter* donné. S'il y a égalité, cette commande   
 retourne un agent pris au hasard parmi ceux ayant cette plus petite   
 valeur. Si vous voulez tous les agents ayant la plus petite valeur,   
 utilisez à la place la commande with-min.

**move-to** La tortue donne à ses coordonnées x et y les mêmes valeurs que   
 celles des coordonnées de l'*agent* spécifié, ce qui la place au même   
 endroit que cet agent. (Si l'agent est un patch, la tortue est déplacée et mise au centre du patch.)

**myself** Signifie « la tortue ou le patch qui m'a demandé de faire ce que je   
 suis justement en train de faire ».  
 Quand il a été demandé à un agent d'exécuter un certain code,   
 l'utilisation de myself dans ce code retourne l'agent (tortue ou patch)   
 qui a fait la demande.  
 myself est le plus souvent utilisé avec of pour lire la valeur d'une   
 variable de l'agent appelant ou donner une valeur cette variable.  
 myself peut être utilisé non seulement dans les blocs de code de la   
 commande ask, mais aussi dans les commandes hatch, sprout, of,   
 with, all?, with-min, with-max, min-one-of, max-one-of, min-n-of et   
 max-n-of.

**other** retourne un ensemble d'agents qui est le même que celui reçu en   
 entrée mais ne contenant plus l'agent appelant.

**reset-ticks** Remets le compteur de cycles (tick counter) à zéro.

**set** Donne à la variable variable la valeur spécifiée dans l'entrée value.

**shape** Variable tortue ou lien préprogrammée.  
 Elle contient une chaîne de caractères qui est le nom de la forme  
 courante de la tortue ou du lien. Vous pouvez changer la valeur de  
 cette variable pour modifier la forme de l'agent.  
 Les nouvelles tortues et les nouveaux liens ont la forme "default"  
 à moins qu'une forme différente n'ait été spécifiée par l'utilisation  
 de la primitive set-default-shape.

**size** variable tortue préprogrammée.  
 Elle contient un nombre qui représente la taille apparente de la  
 tortue. La taille par défaut est 1, ce qui signifie que la taille de la   
 tortue est la même que celle d'un patch.  
 Vous pouvez modifier cette variable pour changer la taille de la  
 tortue.

**tick** Fait avancer le compteur de cycles d'une unité.

**with** Retourne un nouvel ensemble d'agents ne contenant que les agents   
 de l'ensemble agentset ayant satisfait (retourné true (vrai)) à la   
 condition [reporter] donnée. with demande deux entrées : une à   
 gauche, un ensemble d'agents agentset (généralement turtles ou   
 patches), l'autre à droite qui doit être un reporter booléen.

# XX. Références

**Romain Jean**, 2009   
*Guide de la programmation*<https://romainmejean.fr/manuel_netlogo/>

**Carole Adam, Frédéric Amblard, Hélène Arduin, et al.,** 2020  
*CoVprehension,* <https://covprehension.org/>

**Terna P., Pescarmona G., Acquadro A., Pescarmona P., Russo G., Terna S.** (2020), *An Agent-Based Model of the Diffusion of Covid-19 Using NetLogo* <https://terna.to.it/simul/SIsaR.html>

**Jen Badham et al.**, 2020  
Covid-social, <https://github.com/DUCovidTools/covid-social>

1. L’ensemble des primitives peuvent être retrouvées dans le **Dictionnaire NetLogo**, <https://romainmejean.fr/manuel_netlogo/primindex.html>. La liste des principales primitives utilisées dans notre modèle peut être retrouvée en partie **VI. Glossaire** [↑](#footnote-ref-0)