Module 2: Virus Prevention Simulation and Model Inputs

Platform: NetLogo

Following this lesson students should be able to:

1) Describe the factors that can impact a virus's spread throughout a population
2) Modify a given Netlogo model by adding and deleting buttons from the main interface
3) Research the literature to supply model with input parameters from published experimental data
4) Determine what additional rules and/or variables should be added to a model to address its limitations

Purpose: This module modifies the built-in NetLogo Virus code to introduce additional parameters that model an intervention to slow the spread of a virus within a population. Biologically, students will examine **population dynamics and viral spread**. Computationally, students will **research and test the effect of input parameters on a model**.

Biological Terms:

1) Virus: a microorganism that can only reproduce by infecting living cells
2) Infectiousness: the likelihood (probability) that a contagious disease will be transmitted from one person to another
3) Transmission: the act of spreading a disease, like a virus, form organism to another
4) Immunity: the ability of an organism to fight off an infection. After an organism has survived an initial infection of a virus, they often have immunity from that same virus for a long period of time

Computational Terms:

1) Agents: beings in a model that can follow given instructions
2) Turtles: agents in NetLogo that can move around in the world
3) Patches: agents in NetLogo that make up the 2-D space turtles move around on
4) Parameters: a variable included in a model that can be estimated from data
5) Buttons: a feature in NetLogo which allows the user to adjust parameters without manipulating the code

Time Estimation:

1) In-Class Activity: 30 minutes
2) Model Tutorial: 30 minutes
3) Model Testing and Advancement**:** 1 hour

Total: ~2 hours

Module 2: In-Class Activity

Materials: Dice, meter stick, and note cards or stickers to identify different student groups

Rules:

1) Split the students up into 4 groups in an area where they can walk around but not so big that they will be too spread out
2) Let each group decide input parameters for their group:
    a. Define how many people should be in each subgroup to start: Sick-Unmasked ____, Sick-Masked ____, Healthy-Masked ____, Healthy-Unmasked ____
    b. Define how close in feet students need to be for it to be considered "close contact" ____
    c. Define the infection rate for Mask-Mask contact (i.e. roll a two= 1/6) ____ and Mask-Unmask contact (i.e. roll any even number= 1/2) ____ if one of the individuals is sick
3) Have one group perform their simulation at a time while the other groups observe
4) For each simulation, have students walk around randomly and stop them after five seconds
5) If there is a healthy and sick student within the decided distance have each close contact roll a die to determine if the virus will spread
6) Repeat steps 3-4 one more time and tell students who were sick to start with that they are now immune and cannot get sick again
7) Repeat steps 3-5, remembering to tell students they are immune after two cycles until all students are either healthy or immune
8) Repeat steps 4-7 for the remaining three groups
9) Discuss observations in small groups or as a class

Modification for Individual Activity: A single student can cut out small squares of paper, label one side of each square "healthy" (or a smiling face), and the other side "infected" (or a frowning face, virus picture, etc.). Proceed through the steps above.

Suggested Discussion Questions:

1) Which input parameter do you think would have had the greatest effect?
2) Did each group's simulation take about the same about of time to finish? Why or why not?
3) What are some limitations of this model? Would adding/changing any of the input parameters address these limitations?

Part Two: Model Tutorial

1) Open **NetLogo**
2) Under **File** choose **Model Library**
3) Choose **Virus** under **Biology>Evolution**
4) Click **Setup** button
5) Click **Go** button and observe
6) To stop the model, hit **Go** again. This module explains how a virus can spread through a population, but it is not detailed enough to exhibit or test how we could slow or stop the spread of a virus. One proven way to slow the spread of most viruses is to wear masks. This is something Asian countries have been doing for years, but something that did not become popular in the U.S. until the COVID-19 outbreak of 2020
7) Download the **Virus_Mask** text file (Appendix A). This file has been modified to include variables that allow for the testing of how mask compliance in a population can influence the spread of an infectious disease
8) **Copy** the text from the file and **paste** it into the code of the **Virus** module
9) Navigate back to the main interface, you will likely get an error message that says certain variables have not been defined which is what we will do in the next few steps
10) **Right click** on the button that says **Turtle Shape** and **delete** the box
11) Press the **Button** drop down box near the top of the screen and choose a **Slider** to add by clicking on an open area under the sliders already in the interface
12) In the pop-up menu, name the global variable **infectiousness-mask** and change the units to %. Hint: Make sure to type this and later variable names <u>exactly</u> because it directly refers to the code
13) Repeat step 10 but call this global variable **mask-compliance** and also change the units to %
14) Repeat step 10 but call this global variable **sick-people** and also change the units to %. You should now have **7 sliders** (number-people, infectiousness, chance-recover, duration, infectiousness-mask, mask-compliance, and sick-people) which allow you to manually adjust the model input parameters
15) Under **Tools**, choose **Turtles Shapes Editors**
16) Scroll down to **Person** and click **Duplicate**
17) Name the person **mask-person**
18) Use the shape tools to draw a mask on the person and click **Okay** when finished
19) Exit out of the Turtles Shapes Editors
20) Click **Setup**, there should no longer be any errors and many figures should pop up in the black model screen some of whom are healthy (green) and some that are sick (red). The starting amount of people can be adjusted with the slider **number-people**. Some of the people should also be wearing masks, the percentage of people wearing a mask can be adjusted with the **mask-compliance** slider

Part Three: Model Testing and Advancement

1) One important consideration when making a model is to use data from the literature for input parameters to increase its accuracy. Choose a virus and look-up values for **infectiousness** (likelihood of transmission without mask), **infectiousness-mask** (likelihood of transmission with mask), **chance-recover** (survival rate), and **duration** (average length of infection). Good resources for this include: **https://www.cdc.gov/** or **https://pubmed.ncbi.nlm.nih.gov/**
    a. What values did you find?
    Infectiousness:
    Infectiousness mask:
    Chance recover:
    Duration:
2) Incorporate these values into the model by using the sliders
3) Now, make a hypothesis about the percentage of **mask-compliance** that would be necessary to eliminate the virus in **exactly 6 months** (0.5 years) if 25% (**sick-people**) of the population was already infected. Hint: To make observation easier, you may need to adjust the speed of the model with the speed slider at the top of the interface
    a. Hypothesis:
4) Test your prediction by running the model until the infection rate reaches 0%.
    a. Did you ever reach 0% infection?
    b. How long did it take to reach 0% infection?
    c. If it did not take exactly 6 months, try adjusting your prediction for **mask-compliance** and repeat the model test until you get closer to six months. Explain your adjustments and outcomes.
5) The main rule governing this model is that if a sick turtle is on the same patch as another turtle it determines if the other turtle is (1) sick (2) immune and (3) wearing a mask. If the turtle is already sick or immune, nothing happens. If the other turtle is not sick or immune already, it will determine if the turtle is wearing a mask and a random number between 0-100 will be generated and if this number is less than the infectiousness rate set by the user for each scenario, the other turtle will get sick.

The code for this is:

```
to infect ;; turtle procedure
  ask other turtles-here with [ not sick? and not immune? and not mask? ]
    [ if random-float 100 < infectiousness
      [get-sick]]
   ask other turtles-here with [ not sick? and not immune? and mask]
    [ if random-float 100 < infectiousness-mask
      [get-sick]]
end
```

A major limitation of this rule is that it does not consider if the sick turtle is wearing a mask; the infection rate is solely decided by if the other non-sick turtle is wearing a mask.

a. What other parameters and rules would have to be added to the model to address this limitation? Either describe in words or attempt a coded notation of what should be added.

6) This model is extremely basic and has many limitations which make it unlikely to ever to be used to decide an intervention plan
   a. In your opinion, besides the one mentioned above what are the model's three biggest limitations? Hint: you might discuss other interventions to slow a virus's spread besides just wearing masks that the current model fails to consider.

   b. Choose one of these limitations and describe what you think should be added to the model to address this limitation.

Appendix A: Virus_Mask Code

```
turtles-own
  [ sick?              ;; if true, the turtle is infectious
    mask?
    remaining-immunity   ;; how many weeks of immunity the turtle has left
    sick-time          ;; how long, in weeks, the turtle has been infectious
    age ]              ;; how many weeks old the turtle is

globals
  [ %infected          ;; what % of the population is infectious
    %immune            ;; what % of the population is immune
    lifespan           ;; the lifespan of a turtle
    chance-reproduce     ;; the probability of a turtle generating an offspring each tick
    carrying-capacity    ;; the number of turtles that can be in the world at one time
    immunity-duration ]  ;; how many weeks immunity lasts

;; The setup is divided into four procedures
to setup
  clear-all
  setup-constants
  setup-turtles
  update-global-variables
  update-display
  reset-ticks
end

;; We create a variable number of turtles of which a chosen amount are infected and wearing masks
;; and distribute them randomly
to setup-turtles
  create-turtles number-people
```

```
    [ setxy random-xcor random-ycor
      set age random lifespan
      set sick-time 0
      set remaining-immunity 0
      set size 1.5  ;; easier to see
      get-healthy
      no-mask ]
  ask turtles [
  if (random-float 100 < mask-compliance)
    [ wear-mask]]
  ask turtles [
    if (random-float 100 < sick-people)
    [ get-sick]]
end

to get-sick ;; turtle procedure
  set sick? true
  set remaining-immunity 0
end

to wear-mask
   set mask? true
end

to no-mask
  set mask? false
end

to get-healthy ;; turtle procedure
  set sick? false
```

```
  set remaining-immunity 0
  set sick-time 0
end

to become-immune ;; turtle procedure
  set sick? false
  set sick-time 0
  set remaining-immunity immunity-duration
end

;; This sets up basic constants of the model.
to setup-constants
  set lifespan 50 * 52      ;; 50 times 52 weeks = 50 years = 2600 weeks old
  set carrying-capacity 300
  set chance-reproduce 1
  set immunity-duration 52
end

to go
  ask turtles [
    get-older
    move
    if sick? [ recover-or-die ]
    ifelse sick? [ infect ] [ reproduce ]
  ]
  update-global-variables
  update-display
  tick
end
```

```
to update-global-variables
  if count turtles > 0
    [ set %infected (count turtles with [ sick? ] / count turtles) * 100
      set %immune (count turtles with [ immune? ] / count turtles) * 100 ]
end

to update-display
  ask turtles
    [ set shape  ifelse-value mask? ["mask-person"]["person"]
      set color ifelse-value sick? [ red ] [ ifelse-value immune? [ grey ] [ green ] ] ]
end

;;Turtle counting variables are advanced.
to get-older ;; turtle procedure
  ;; Turtles die of old age once their age exceeds the
  ;; lifespan (set at 50 years in this model).
  set age age + 1
  if age > lifespan [ die ]
  if immune? [ set remaining-immunity remaining-immunity - 1 ]
  if sick? [ set sick-time sick-time + 1 ]
end

;; Turtles move about at random.
to move ;; turtle procedure
  rt random 100
  lt random 100
  fd 1
end

;; If a turtle is sick, it infects other turtles on the same patch.
```

```
;; Immune turtles don't get sick.
to infect ;; turtle procedure
  ask other turtles-here with [ not sick? and not immune? and not mask? ]
    [ if random-float 100 < infectiousness
      [get-sick]]
   ask other turtles-here with [ not sick? and not immune? and mask?]
    [ if random-float 100 < infectiousness-mask
      [get-sick]]
end

;; Once the turtle has been sick long enough, it
;; either recovers (and becomes immune) or it dies.
to recover-or-die ;; turtle procedure
  if sick-time > duration                    ;; If the turtle has survived past the virus' duration, then
    [ ifelse random-float 100 < chance-recover   ;; either recover or die
      [ become-immune ]
      [ die ] ]
end

;; If there are less turtles than the carrying-capacity
;; then turtles can reproduce.
to reproduce
  if count turtles < carrying-capacity and random-float 100 < chance-reproduce
    [ hatch 1
      [ set age 1
        lt 45 fd 1
        get-healthy ] ]
end

to-report immune?
```

```
  report remaining-immunity > 0
end

to startup
  setup-constants ;; so that carrying-capacity can be used as upper bound of number-people slider
end
```