IDENTITIES

Enlightening Interdisciplinarity in STEM for Teaching

Simulations of complex systems

Co-funded by the Erasmus+ Programme of the European Union

Grant Agreement n°2019-1-IT02-KA203-063184

Some definitions

A simulation is a computer-implemented method for exploring the properties of a mathematical model where analytical methods are not available (Humphreys, 1991)

Some definitions

- A simulation is a computer-implemented method for exploring the properties of a mathematical model where analytical methods are not available (Humphreys, 1991)

Simulations are understood to be purely computational

Exploratory purpose

Excludes simulations for which analytical methods are available

Some definitions

A simulation is a computer-implemented method for exploring the properties of a mathematical model where analytical methods are not available (Humphreys, 1991)

A simulation imitates a process by means of another process (Hartmann, 1996)

Some definitions

- A simulation is a computer-implemented method for exploring the properties of a mathematical model where analytical methods are not available (Humphreys, 1991)

Simulations are understood to be purely computational

Exploratory role

Excludes simulations for which analytical methods are available

- A simulation imitates a process by means of another process (Hartmann, 1996)

Imitative purpose

Simulation need not be computational

Excludes simulations that use a model to present the structure (not the dynamics) of a system

Some definitions

A simulation is a computer-implemented method for exploring the properties of a mathematical model where analytical methods are not available (Humphreys, 1991)

A simulation imitates a process by means of another process (Hartmann, 1996)

A system S provides a *core simulation of an* object or process B if and only if S is a concrete computational device that produces, via a temporal process, solutions for a computational model that correctly represents B, either dynamically or statically. If, in addition, the computational model used by S correctly represents the structure of a real system R, then S provides a *core simulation* of a system R with respect to B (Humphreys, 2004)

A possible definition of simulation

A system S is a simulation of an object or process M if and only if S is a concrete computational device that produces, by means of a temporal process, solutions for a computational model that correctly represents M.

If, in addition, the computational model used by S correctly represents the structure of a real system R, then S provides a simulation of an R system with respect to the M model.

- R — the real system
- M — the object or process model
- Mc — the computational model
- S — the simulation

(from Humphreys, 2004)

A possible definition of simulation

A system S is a simulation of an object or process M if and only if S is a concrete computational device that produces, by means of a temporal process, solutions for a computational model that correctly represents M.

If, in addition, the computational model used by S correctly represents the structure of a real system R, then S provides a simulation of an R system with respect to the M model.

The correctness of a simulation depends on two factors

- The adherence of the simulation to the **model**
- The adherence of simulation to the real **world**

A possible definition of simulation

A system S is a simulation of an object or process M if and only if S is a concrete computational device that produces, by means of a temporal process, solutions for a computational model that correctly represents M.

If, in addition, the computational model used by S correctly represents the structure of a real system R, then S provides a simulation of an R system with respect to the M model.

- According to this definition, the purpose of a simulation is to solve a computational model, find solutions to it
- The solution is derived in a time-based, **step-by-step** process
- Simulation is of particular interest **when analytical methods are not available**
- This is the case with complex systems

Uses of simulations

- In science, simulations are used for many purposes (Grüne-Yanoff & Weirich, 2010):
    - Prediction of a future event/behaviour (within a certain probability)
    - Explanation of a concrete phenomenon, showing its history, identifying the causal relationships that produced it
    - Decision-making in contexts of uncertainty and complexity

Two examples

of simulations of complex systems

Example 1: Lotka-Volterra

Model of Lotka-Volterra

- Describes the dynamics of complex biological systems in which two species interact one as prey, the other as predator (Volterra, 1926)
- Assumptions on which the model is based:
    - Prey has unlimited food
    - Predators' only source of livelihood is prey
    - Prey only die a natural death
    - No evolutionary mechanisms are in place
    - The external environment does not change in favour of any of the species

x(t) = number of prey at time t
dx(t)/dt = rate of change of the prey population over time
y(t) = number of predators at time t
dy(t)/dt = rate of change of the predator population over time t

$$\frac{dx(t)}{dt} = ax(t) - by(t)x(t)$$

$$\frac{dy(t)}{dt} = cx(t)y(t) - dy(t)$$

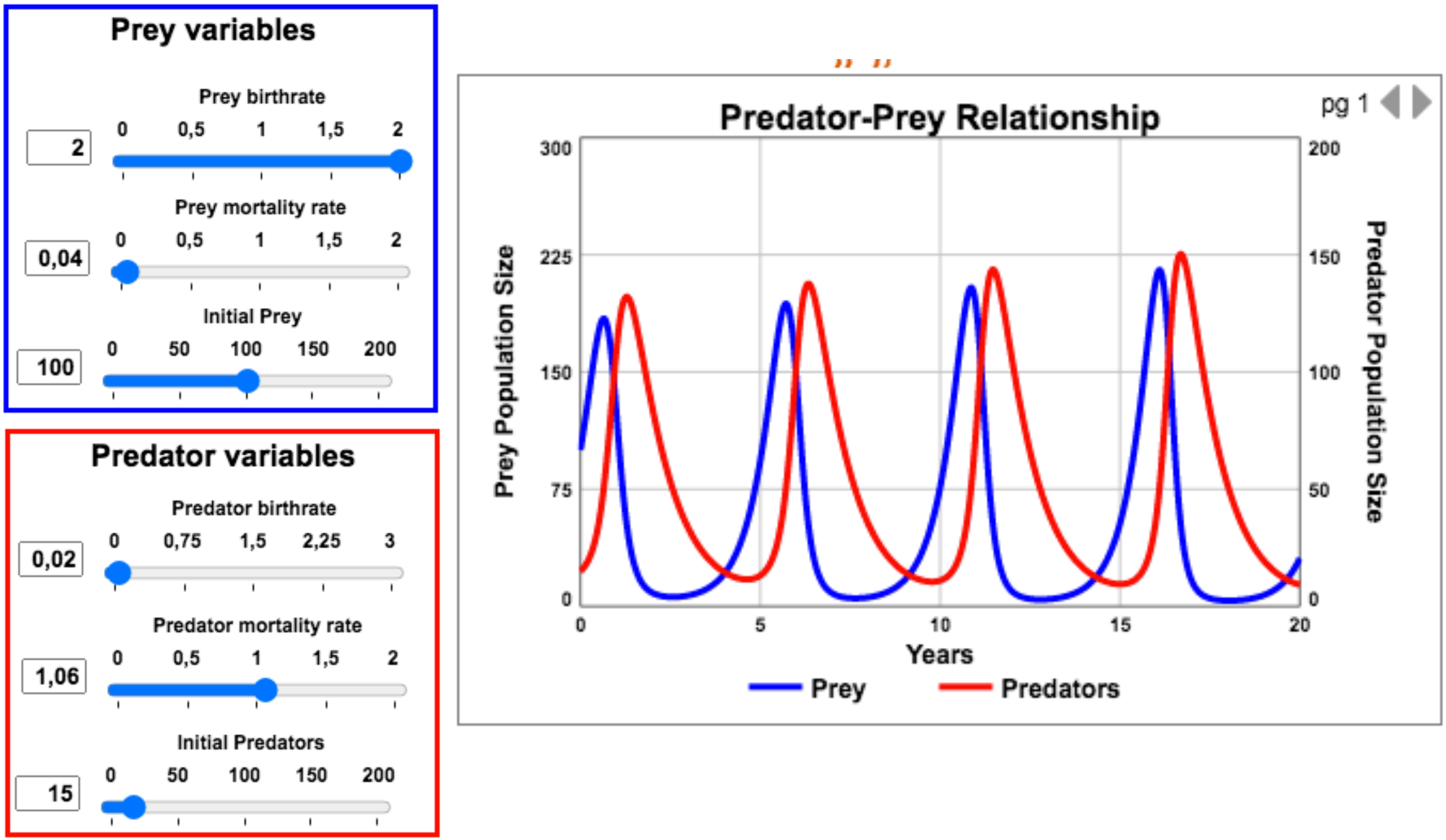
Coefficients:
a = birth of prey
b = predation
c = encounter between prey and predators
d = natural death of predators

From model to simulation

- The model is expressed as a system of two differential equations
- In this particular case, it is possible to solve the model analytically, i.e. one can express $x(t)$ or $y(t)$ as functions of t, $x(0)$ and $y(0)$, but this route is not always possible for complex dynamic systems
- Another way is to use simulation, which makes use of numerical integration methods

From model to simulation...

Lotka-Volterra simulation



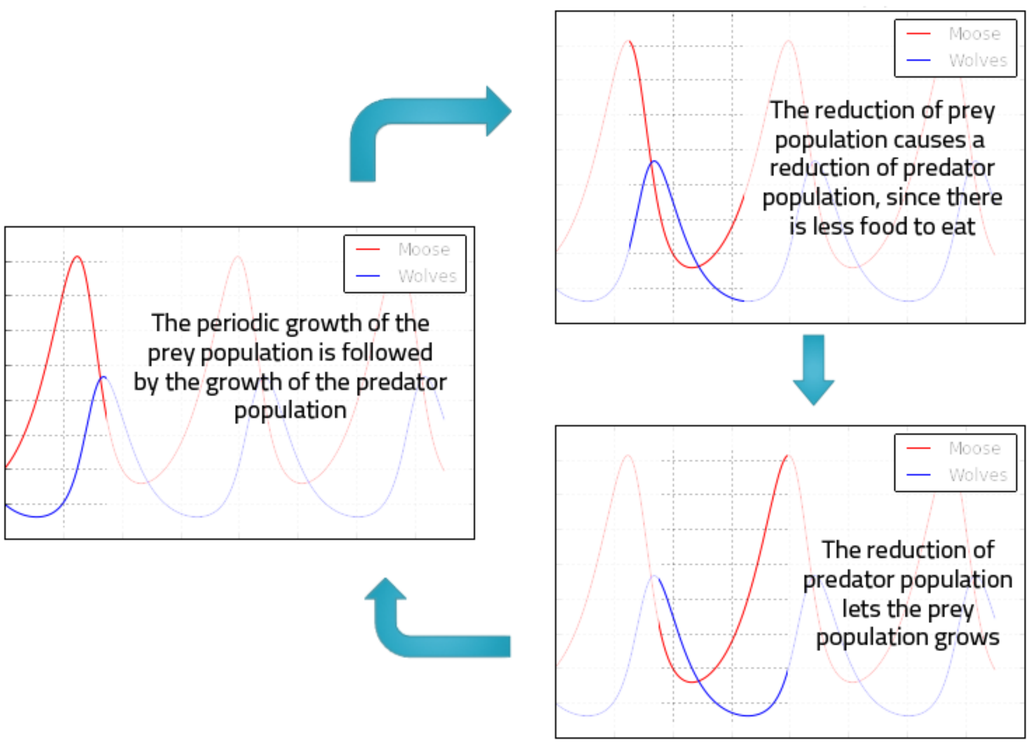
https://sites.google.com/site/biologydarkow/ecology/predator-prey-simulation-of-the-lotka-volterra-model

Lotka-Volterra simulation

https://sites.google.com/site/biologydarkow/ecology/predator-prey-simulation-of-the-lotka-volterra-model

An example of negative feedback!

- Interconnections between predator and prey populations give rise to negative feedback
- This simple interaction allows the system to self-regulate and remain in balance!



Example 2: Schelling

Schelling model

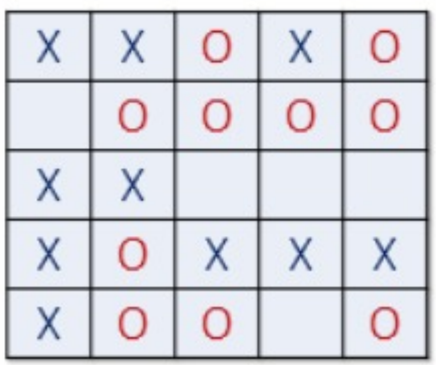
- In this model (Schelling, 1969), an environment consists of two types of individuals
- At first they are placed randomly on a grid
- At each step, an individual moves from his position if there are more than 70% different individuals among his neighbours

X	X	O	X	O
	O	O	O	O
X	X			
X	O	X	X	X
X	O	O		O

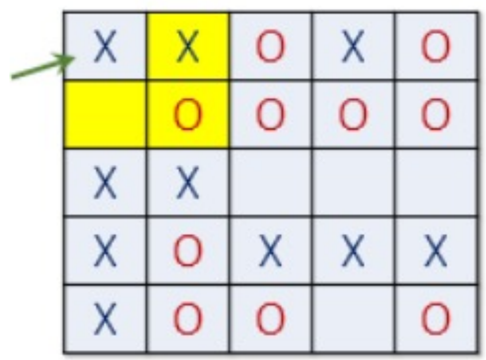
Agents placed randomly on a grid

Schelling model

- In this model (Schelling, 1969), an environment consists of two types of individuals
- At first they are placed randomly on a grid
- At each step, an individual moves from his position if there are more than 70% different individuals among his neighbours



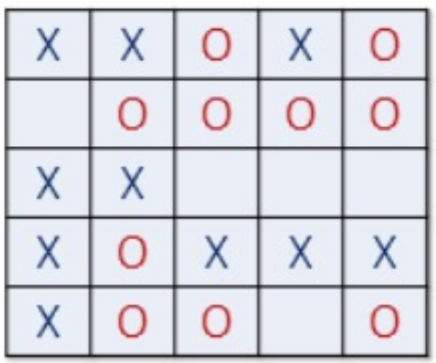
Agents placed randomly on a grid



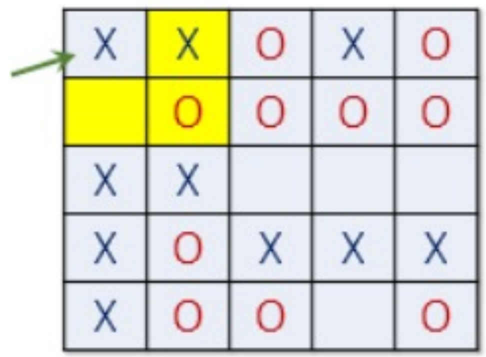
x satisfied because 1 in 2 (50% < 70%) of its neighbours is either

Schelling model

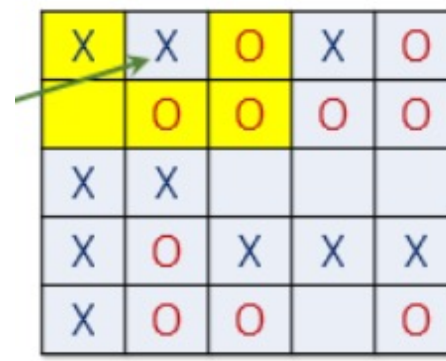
- In this model (Schelling, 1969), an environment consists of two types of individuals
- At first they are placed randomly on a grid
- At each step, an individual moves from his position if there are more than 70% different individuals among his neighbours



Agents placed randomly on a grid

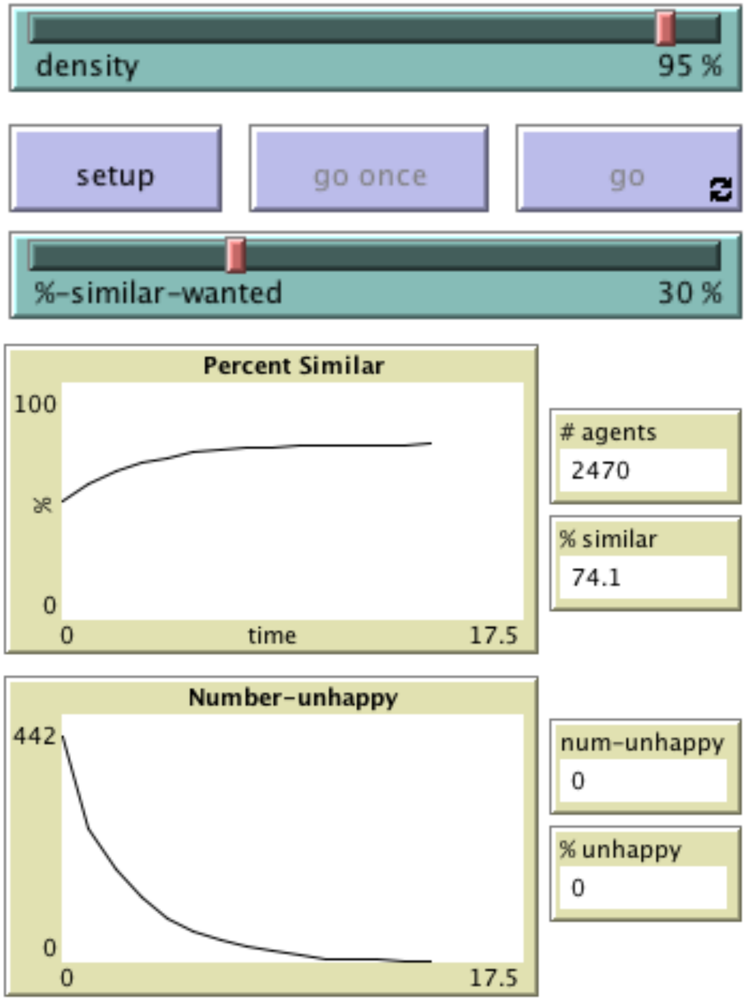


x satisfied because 1 in 2 (50% < 70%) of its neighbours is either

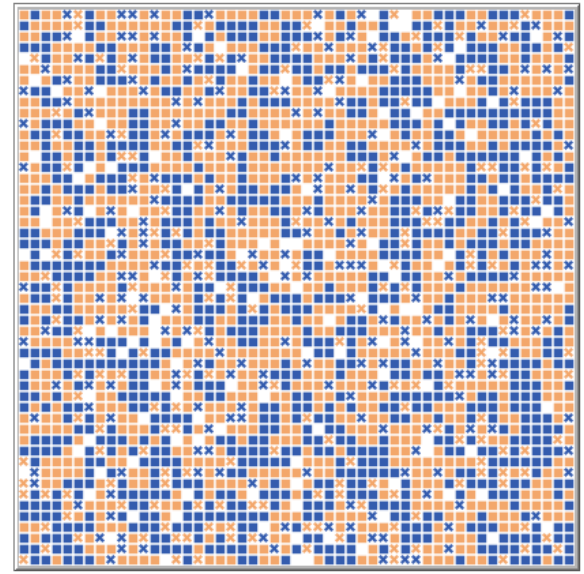


x dissatisfied because 3 out of 4 (75% > 70%) of his neighbours are either

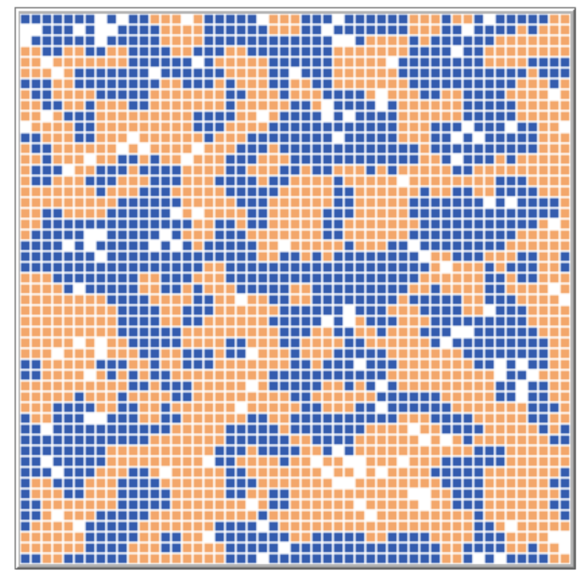
Schelling simulation



Initial situation: mixing



Final situation: segregation



https://www.netlogoweb.org/launch#https://www.netlogoweb.org/assets/modelslib/Sample%20Models/Social%20Science/Segregation.nlogo

Let's look 'inside' a NetLogo simulation

```
globals [
  percent-similar

  percent-unhappy
]

turtles-own [
  happy?

  similar-nearby
  other-nearby
  total-nearby
]
```

It defines new global variables. Global variables are 'global' because they are <u>accessible by all agents and can be used anywhere in a model</u>.

Most often, globals is used to define variables or constants that need to be used in many parts of the program.

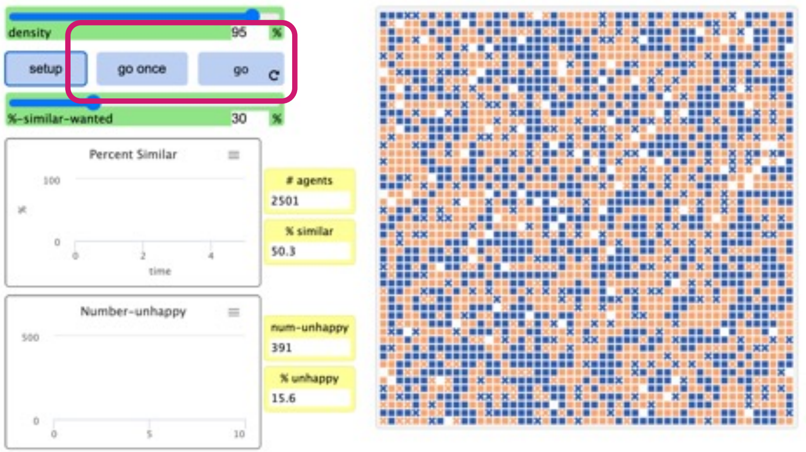
The turtles-own keyword can only be used at the beginning of a program, before any function definitions. It defines the variables belonging to each turtle.

```
globals [
  percent-similar  ; on the average, what percent of a turtle's neighbors
                   ; are the same color as that turtle?
  percent-unhappy  ; what percent of the turtles are unhappy?
]

turtles-own [
  happy?           ; for each turtle, indicates whether at least %-similar-wanted percent of
                   ;   that turtle's neighbors are the same color as the turtle
  similar-nearby   ; how many neighboring patches have a turtle with my color?
  other-nearby     ; how many have a turtle of another color?
  total-nearby     ; sum of previous two variables
]
```

```
to setup
  clear-all
  ; create turtles on random patches.
  ask patches [

    set pcolor white
    if random 100 < density [    ; set the occupancy density
      sprout 1 [
        ; 105 is the color number for "blue"
        ; 27 is the color number for "orange"
        set color one-of [105 27]
        set size 1
      ]
    ]
  ]
  update-turtles
  update-globals
  reset-ticks
end
```

The setup procedure is the one that is run when the setup button is pressed.

```
; run the model for one tick
to go
  if all? turtles [ happy? ] [ stop ]
  move-unhappy-turtles
  update-turtles
  update-globals
  tick
end
```

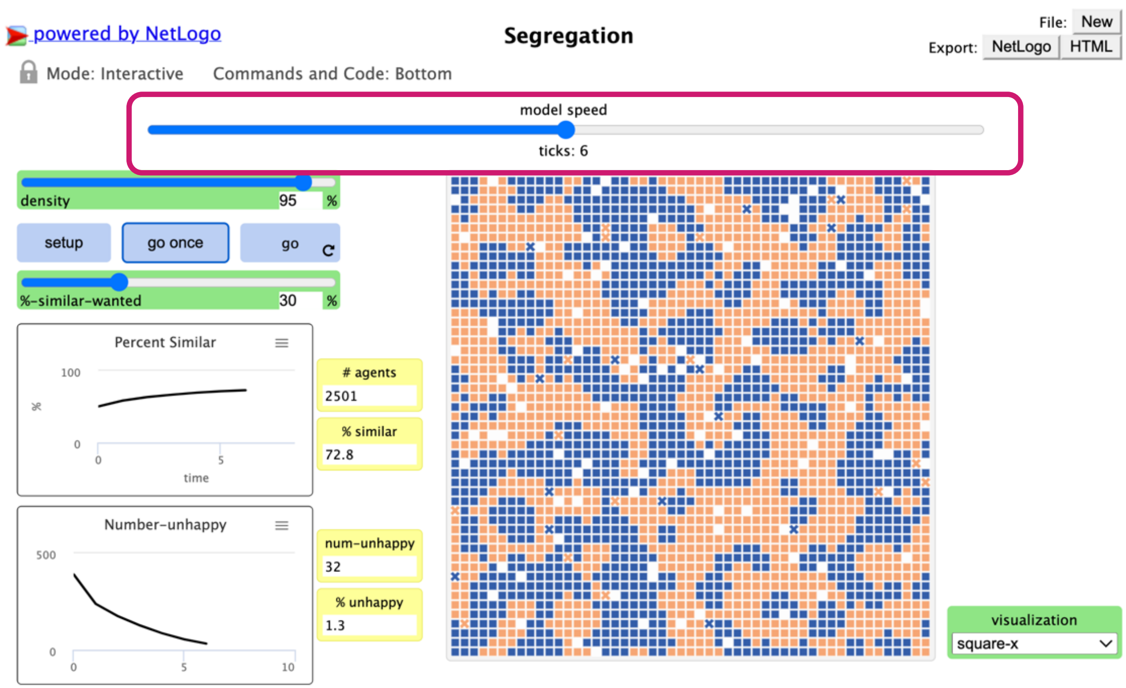


The go procedure is the one that is run when the go button is pressed.

```
; run the model for one tick
to go
  if all? turtles [ happy? ] [ stop ]
  move-unhappy-turtles
  update-turtles
  update-globals
  tick
end
```

Advances the tick counter by one and updates all plots. If the tick counter has not been started yet with reset-ticks, an error results (see setup).

Normally tick goes at the end of a go procedure.



```
; run the model for one tick
to go
  if all? turtles [ happy? ] [ stop ]
  move-unhappy-turtles
  update-turtles
  update-globals
  tick
end
```

```
; unhappy turtles try a new spot
to move-unhappy-turtles
  ask turtles with [ not happy? ]
    [ find-new-spot ]
end
```

```
; move until we find an unoccupied spot
to find-new-spot
  rt random-float 360
  fd random-float 10
  if any? other turtles-here [ find-new-spot ] ; keep going until we
  move-to patch-here  ; move to center of patch
end
```

```
; run the model for one tick
to go
  if all? turtles [ happy? ] [ stop ]
  move-unhappy-turtles
  update-turtles
  update-globals
  tick
end
```

```
to update-turtles
  ask turtles [
    ; in next two lines, we use "neighbors" to test the eight patches
    ; surrounding the current patch
    set similar-nearby count (turtles-on neighbors)  with [ color = [ color ] of myself ]
    set other-nearby count (turtles-on neighbors) with [ color != [ color ] of myself ]
    set total-nearby similar-nearby + other-nearby
    set happy? similar-nearby >= (%-similar-wanted * total-nearby / 100)
    ; add visualization here
    if visualization = "old" [ set shape "default" set size 1.3 ]
    if visualization = "square-x" [
      ifelse happy? [ set shape "square" ] [ set shape "X" ]
    ]
  ]
end
```

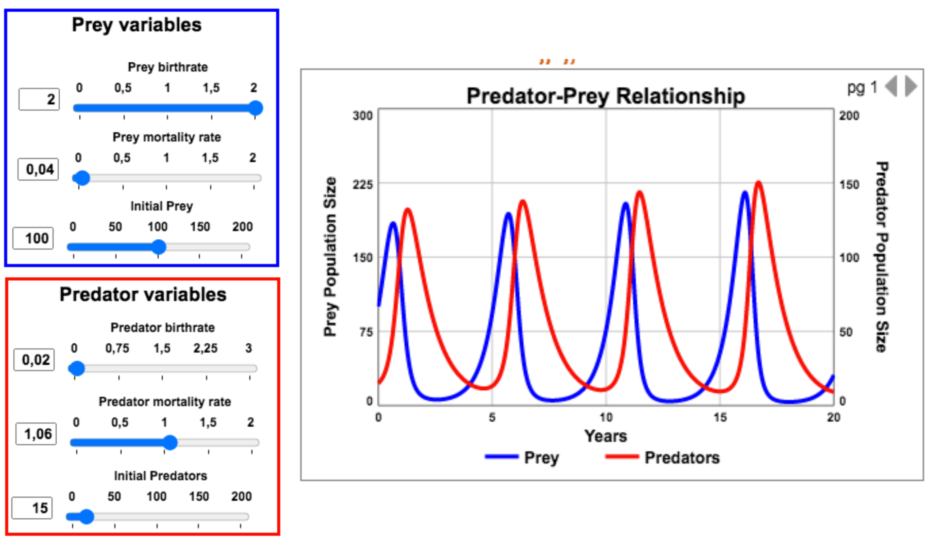
```
; run the model for one tick
to go
  if all? turtles [ happy? ] [ stop ]
  move-unhappy-turtles
  update-turtles
  update-globals
  tick
end
```

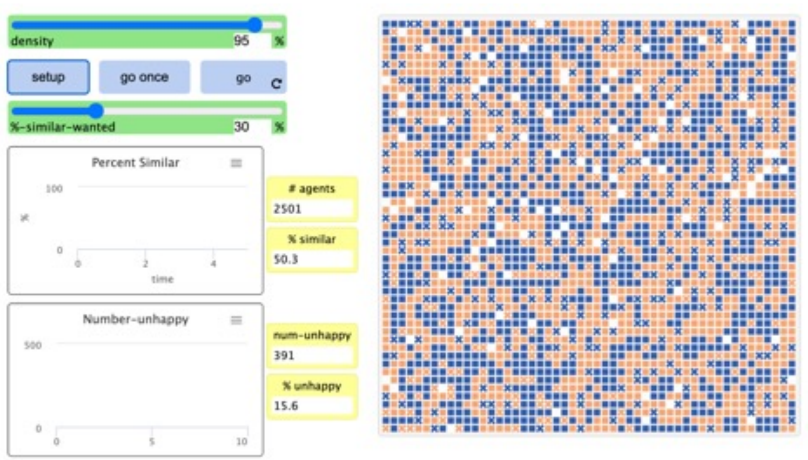
```
to update-globals
  let similar-neighbors sum [ similar-nearby ] of turtles
  let total-neighbors sum [ total-nearby ] of turtles
  set percent-similar (similar-neighbors / total-neighbors) * 100
  set percent-unhappy (count turtles with [ not happy? ]) / (count turtles) * 100
end
```

An example of an emerging property!

- An initially mixed population gives rise to an environment in which there are separate groups of individuals of the same category
- An albeit weak preference of individuals is sufficient to cause a segregation of the two types of individuals
- Segregation is an emergent property because:
    - It is typical of the macroscopic scale of the system
    - It is not due to any centralised control or explicit decision of the two groups as a whole
    - There is no direct causal link between the rules on individuals (microscopic level) and the aggregate result of the evolution of the system macroscopic level)

What differences between the two models?





Equation-based simulations

- In these simulations, the dynamics of a target system is described by means of differential equations that, when solved numerically, allow the future state of the system to be derived from the present state
- Variables related to the macroscopic system appear in the equations
- The target system is modelled as an undifferentiated 'whole'.

Agent-based simulations

- In these simulations, the dynamics of a target system are generated by making individual agents evolve according to their own rules of behaviour
- A description of the macroscopic properties of the system that 'emerge' as a result of running the simulation is missing
- The system is modelled as a group of individuals/agents

Equation-based and agent-based simulations in educational research

- From the conceptual and disciplinary distinction between these two approaches...
- ... to the characterisation of students' forms of reasoning about complex phenomena (Jacobson & Wilensky, 2006)

Macro reasoning

- System as population
- Focus on macroscopic properties

Micro reasoning

- System as a set of individual components
- Interaction between components is determined by rules

Both are essential for a deep understanding of the complexity of systems (**complementarity** between macro and micro reasoning) (Stroup & Wilensky, 2014)

Student difficulties

- Distinguishing the 'levels' of which the system is composed
- Formulating explanations of complex phenomena: tendency to *slippage* from the microscopic level of parts to the macroscopic level of systemic behaviour
- Recognising causal relationships within the system

Need to build
dynamic relationships
between these two
planes of reasoning

$$\frac{dx(t)}{dt} = ax(t) - by(t)x(t)$$

$$\frac{dy(t)}{dt} = cx(t)y(t) - dy(t)$$

Let us try to generate an agent-based simulation equivalent for the Lotka-Volterra model.

How would you proceed?

Different rules for wolves and sheep

Wolf:

- He starts his life with a random amount of energy
- With each tick of the simulation, it moves to an adjacent cell and its energy decreases by E_1
- If a sheep is in the same box, it eats it and its energy increases by E_2
- When the energy reaches 0, the wolf dies
- At any given time, it has a probability R_1 of reproducing itself

Sheep:

- With each tick, it moves to an adjacent cell
- At any given time, it has a probability R_2 of reproducing itself

Differences between equation-based and agent-based models

$$\frac{dx(t)}{dt} = ax(t) - by(t)x(t)$$

$$\frac{dy(t)}{dt} = cx(t)y(t) - dy(t)$$

x(t) = number of prey at time t
dx(t)/dt = rate of change of the prey population over time
y(t) = number of predators at time t
dy(t)/dt = rate of change of the predator population over time t

Coefficients:
a = birth frequency of prey
b = predation frequency
c = birth frequency of predators
d = frequency of natural death of predators

Wolf:

- He starts his life with a random amount of energy
- With each tick of the simulation, it moves to an adjacent cell and its energy decreases by E_1
- If a sheep is in the same box, it eats it and its energy increases by E_2
- When the energy reaches 0, the wolf dies
- At any given time, it has a probability R_1 of reproducing itself

Sheep:

- With each tick, it moves to an adjacent cell
- At any given time, it has a probability R_2 of reproducing itself

Differences between equation-based and agent-based models

$$\frac{dx(t)}{dt} = ax(t) - by(t)x(t)$$

$$\frac{dy(t)}{dt} = cx(t)y(t) - dy(t)$$

x(t) = **number of prey** at time t
dx(t)/dt = rate of change of the prey population over time
y(t) = **number of predators** at time t
dy(t)/dt = rate of change of the predator population over time t

Coefficients:
a = birth frequency of prey
b = predation frequency
c = birth frequency of predators
d = frequency of natural death of predators

(every single) Wolf:

- He starts his life with a random amount of energy
- With each tick of the simulation, it moves to an adjacent cell and its energy decreases by E_1
- If a sheep is in the same box, it eats it and its energy increases by E_2
- When the energy reaches 0, the wolf dies
- At any given time, it has a probability R_1 of reproducing itself

(every single) sheep:

- With each tick, it moves to an adjacent cell
- At any given time, it has a probability R_2 of reproducing itself

Differences between equation-based and agent-based models

$$\frac{dx(t)}{dt} = ax(t) - by(t)x(t)$$

x(t
dx

y(t
dy

Coefficients:
a = birth frequency of prey
b = predation frequency
c = birth frequency of predators
d = frequency of natural death of predators

(every single) Wolf:

- He starts his life with a random amount of energy

cell

- At any given time, it has a probability R_2 of reproducing itself

In the equation model, only macroscopic quantities appear, whereas in the agent model, the focus is on the individual to whom the rules are associated

Differences between equation-based and agent-based models

$$\frac{dx(t)}{dt} = ax(t) - by(t)x(t)$$

$$\frac{dy(t)}{dt} = cx(t)y(t) - dy(t)$$

x(t) = number of prey at time t
dx(t)/dt = rate of change of the prey population over time
y(t) = number of predators at time t
dy(t)/dt = rate of change of the predator population over time t

Coefficients:
a = birth **frequency of** prey
b = predation **frequency**
c = birth **frequency of** predators
d = **frequency of** natural death of predators

Wolf:

- He starts his life with a **random** amount of energy
- With each tick of the simulation, it **randomly** moves to an adjacent cell and its energy decreases by E_1
- If a sheep is in the same box, it eats it and its energy increases by E_2
- When the energy reaches 0, the wolf dies
- At any given time, it has a **probability** R_1 of reproducing itself

Sheep:

- With each tick, it moves to an adjacent cell
- At any given time, it has a **probability** R_2 of reproducing itself

Differences between equation-based and agent-based models

$$\frac{dx(t)}{dt} = ax(t) - by(t)x(t)$$

$$\frac{dy(t)}{dt} = cx(t)y(t) - dy(t)$$

x(t

dx

y(t

dy

population over time t

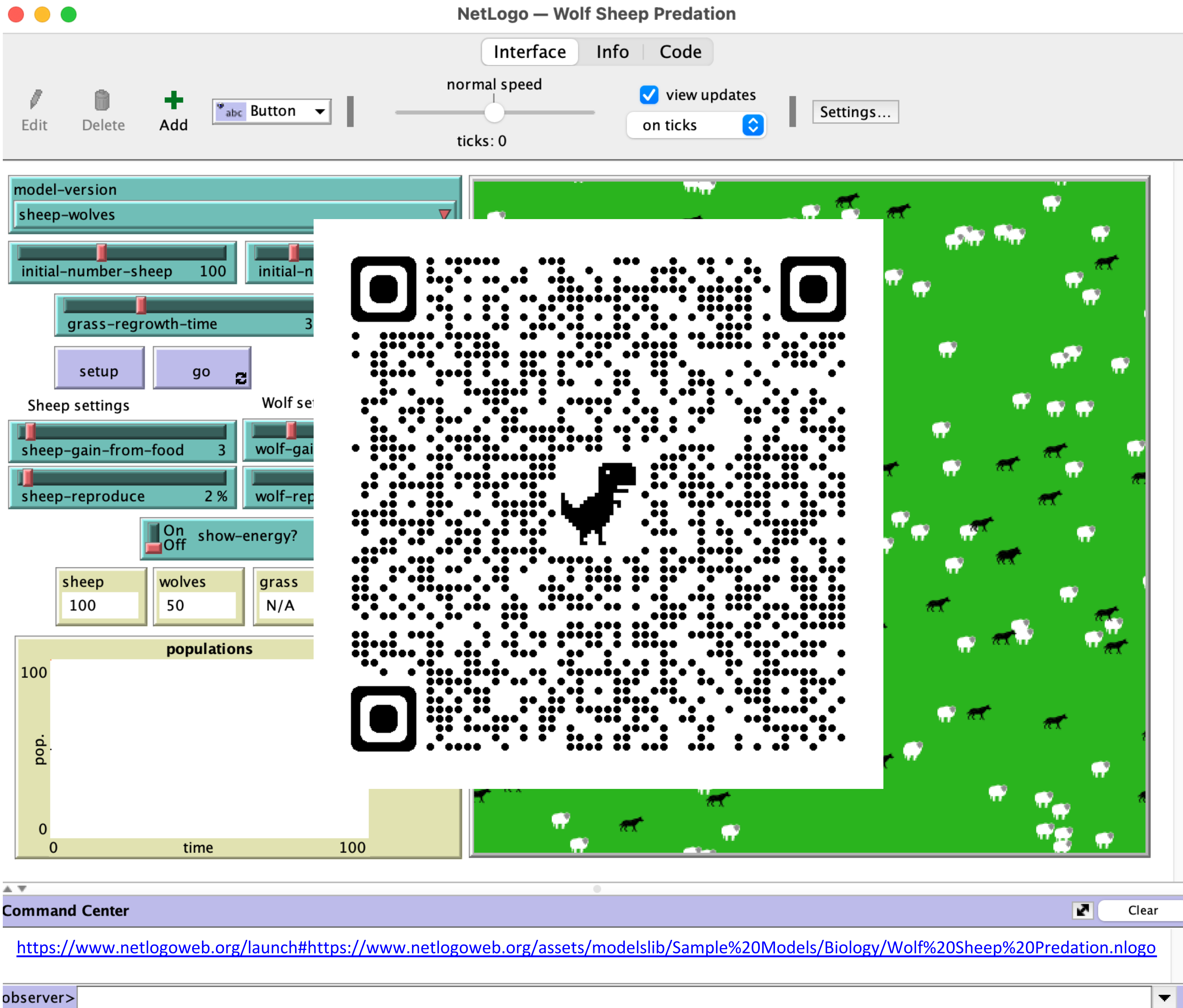
Coefficients:
a = birth **frequency of** prey
b = predation **frequency**
c = birth **frequency of** predators
d = **frequency of** natural death of predators

Wolf:

- He starts his life with a **random** amount of energy
- With each tick of the simulation, it **randomly** moves to an adjacent cell and its energy decreases by E_1

- With each tick, it moves to an adjacent cell
- At any given time, it has a **probability** R_2 of reproducing itself

The equation model is deterministic while the agent model is probabilistic


NetLogo — Wolf Sheep Predation
Interface
Info
Code
Edit
Delete
Add
Button
normal speed
ticks: 0
view updates
on ticks
Settings...
model-version
sheep-wolves
initial-number-sheep 100
grass-regrowth-time
setup
go
Sheep settings
sheep-gain-from-food 3
sheep-reproduce 2 %
On
Off
show-energy?
sheep
100
wolves
50
grass
N/A
populations
100
pop.
0
0
time
100
Command Center
Clear
https://www.netlogoweb.org/launch#https://www.netlogoweb.org/assets/modelslib/Sample%20Models/Biology/Wolf%20Sheep%20Predation.nlogo
observer>

Where do we find these rules in the simulation?

```
ask sheep [
  move
  reproduce-sheep
]
```

```
to move  ; turtle procedure
  rt random 50
  lt random 50
  fd 1
end
```

```
to reproduce-sheep  ; sheep procedure
  if random-float 100 < sheep-reproduce [  ; throw "dice" to see if you will reproduce
    hatch 1 [ rt random-float 360 fd 1 ]   ; hatch an offspring and move it forward 1 step
  ]
end
```

Where do we find these rules in the simulation?

```
ask wolves [
  move
  set energy energy - 1
  eat-sheep
  death
  reproduce-wolves
]
```

```
to move  ; turtle procedure
  rt random 50
  lt random 50
  fd 1
end
```

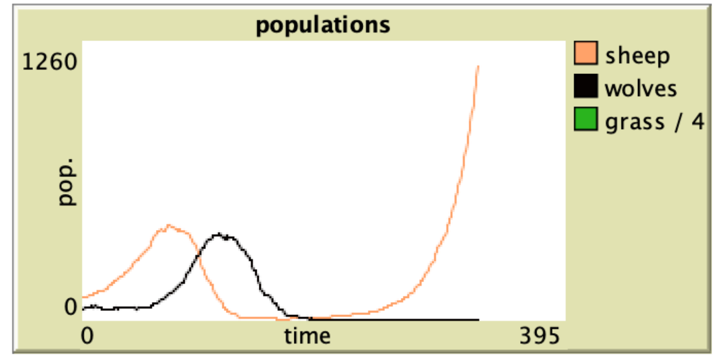
```
to eat-sheep  ; wolf procedure
  let prey one-of sheep-here
  if prey != nobody  [
    ask prey [ die ]
    set energy energy + wolf-gain-from-food
  ]
end
```

```
to death  ; turtle procedure
  ; when energy dips below zero, die
  if energy < 0 [ die ]
end
```

```
to reproduce-wolves  ; wolf procedure
  if random-float 100 < wolf-reproduce [
    set energy (energy / 2)
    hatch 1 [ rt random-float 360 fd 1 ]
  ]
end
```

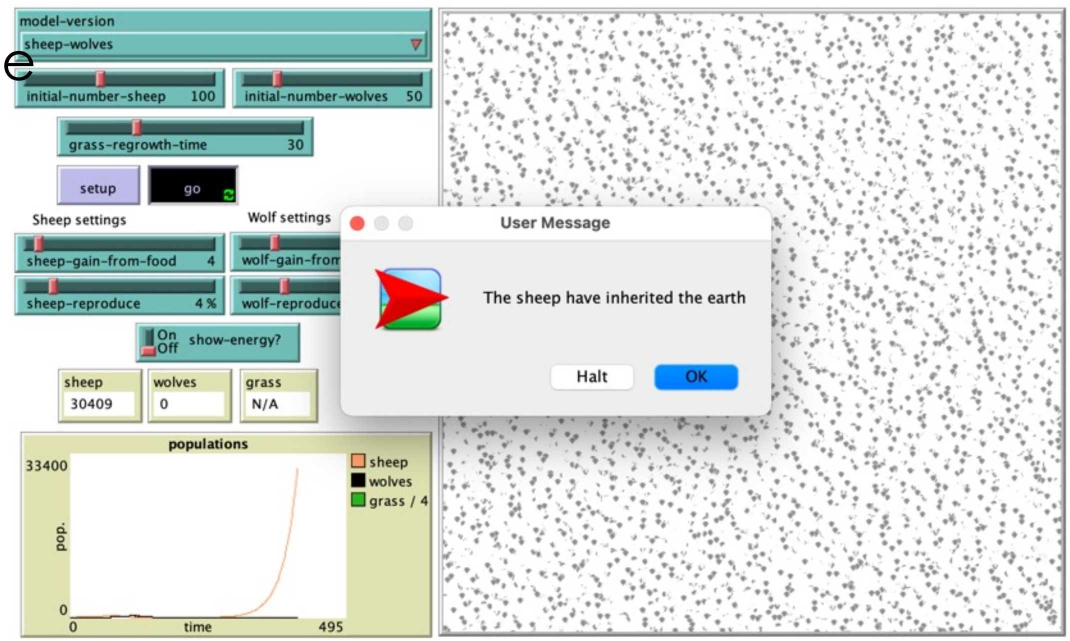
What happens to the system with these rules?

- The rules of the model reproduce the oscillation of the two populations we observed earlier



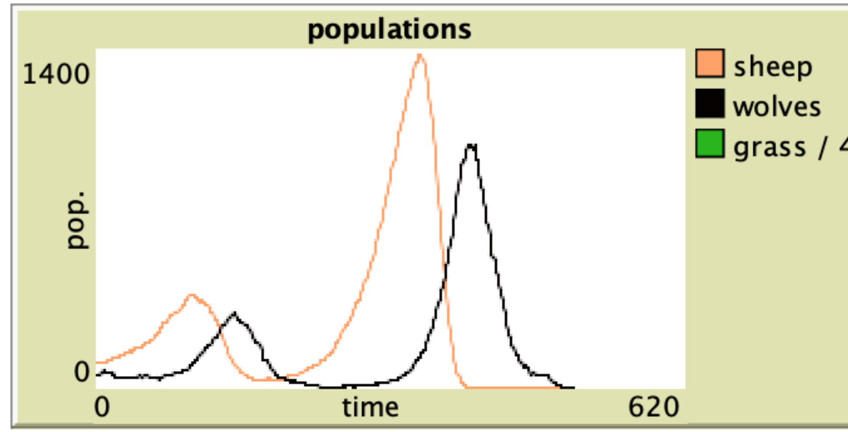
- This pattern, however, is only transient and unstable

- Or the number of sheep "explodes" by increasing exponentially...



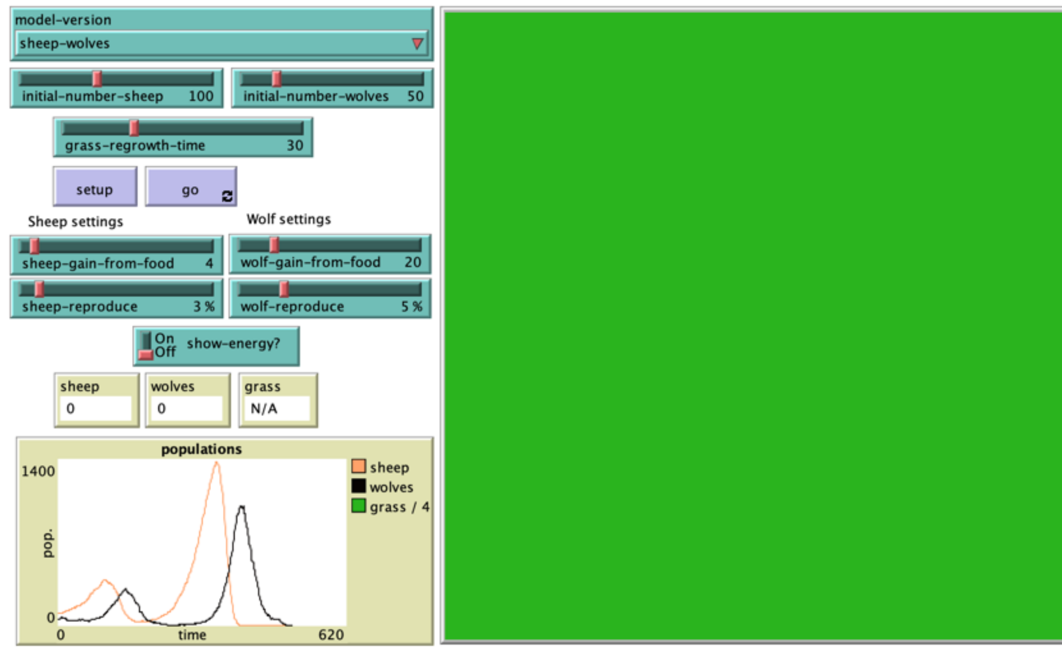
What happens to the system with these rules?

- Or both populations end up going extinct after a few swings…



- Why does this happen?

- What should we fix in our model?



Actually, the behaviour observed with the simulation is not so absurd...

- In 1934, the first **laboratory** experiments on interactions between prey and predators found the same result!

EXPERIMENTAL ANALYSIS OF VITO VOLTERRA'S MATHEMATICAL THEORY OF THE STRUGGLE FOR EXISTENCE

In the last four years I have carried on an experimental investigation of the processes of the struggle for existence among unicellular organisms. Experiments on the competition between two species for a common place in the microcosm agreed completely with Volterra's theoretical equations, but as regards the processes of one species devouring another our results are not concordant with the forecasts of the mathematical theory. All this extensive experimental

- Either the predators would eat all the prey and then starve to death, or the predators would die first and the prey would multiply disproportionately

- What is similar between laboratory and simulation, and different from the natural environment?
- What is similar between laboratory and simulation, and different from the assumptions of the mathematical model?

Laboratory vs. natural environment

- In the laboratory, there are no limits to prey population growth
- In nature yes (finite resources and density)

- In the laboratory, there is no 'environmental complexity': prey cannot escape predators by taking refuge in a certain part of the environment

- Our agent simulation also experiences the same limitations, so **we reproduced the behaviour of the experiment**

A struggle between models

- Yet the Lotka-Volterra model also had the same assumptions but the solutions of the equations are stable and reproduce the behaviour in nature, but not that in the laboratory!

- Is it the Lotka-Volterra model to be 'false' or the agent?

- Compared to agent simulations, classical equation models make it very easy to obtain implausible macroscopic results!

A struggle between models

- This happens because by modelling to equation, I focus on what **I want to** achieve (stability)

- Whereas in the agent approach, I am only concerned with assigning rules, without any assumptions at macro leve

- It is therefore possible that there are simply **NO** rules at the local level that produce the result of stability that the equation model predicted!

COGNITION AND INSTRUCTION, *24*(2), 171–209

Thinking Like a Wolf, a Sheep, or a Firefly: Learning Biology Through Constructing and Testing Computational Theories— An Embodied Modeling Approach

Uri Wilensky
Center for Connected Learning and Computer-Based Modeling
Departments of Learning Sciences and Computer Sciences
Northwestern University

Kenneth Reisman
Stanford University

Differences between equation-based and agent-based models

$$\frac{dx(t)}{dt} = ax(t) - by(t)x(t)$$

Wolf:

- He starts his life with a random amount of energy
- With each tick of the simulation, it moves to

The equation model contains a pre-judgement on the system (top-down), whereas the agent model produces everything from the rules (bottom-up)

cell

- At any given time, it has a probability R_2 of reproducing itself

Coefficients:
a = birth frequency of prey
b = predation frequency
c = birth frequency of predators
d = frequency of natural death of predators

Let's try some new rules

Wolf:

- He starts his life with a random amount of energy
- With each tick of the simulation, it moves to an adjacent cell and its energy decreases by E_1
- If a sheep is in the same box, it eats it and its energy increases by E_2
- When the energy reaches 0, the wolf dies
- At any given time, it has a probability R_1 of reproducing itself

Sheep:

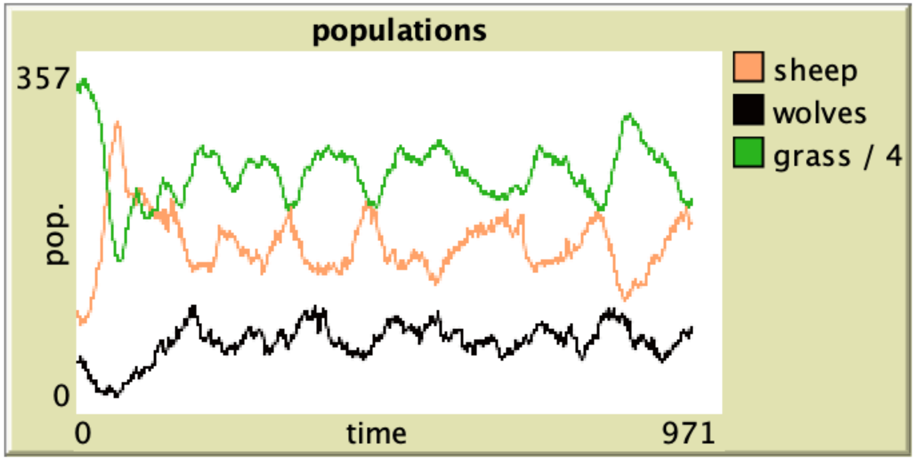
- He starts his life with a random amount of energy
- With each tick, it moves to an adjacent cell and its energy decreases by E_3
- If grass is found in the same box, it eats it and its energy increases by E4
- When the energy reaches 0, the sheep dies
- At any given time, it has a probability R_2 of reproducing itself

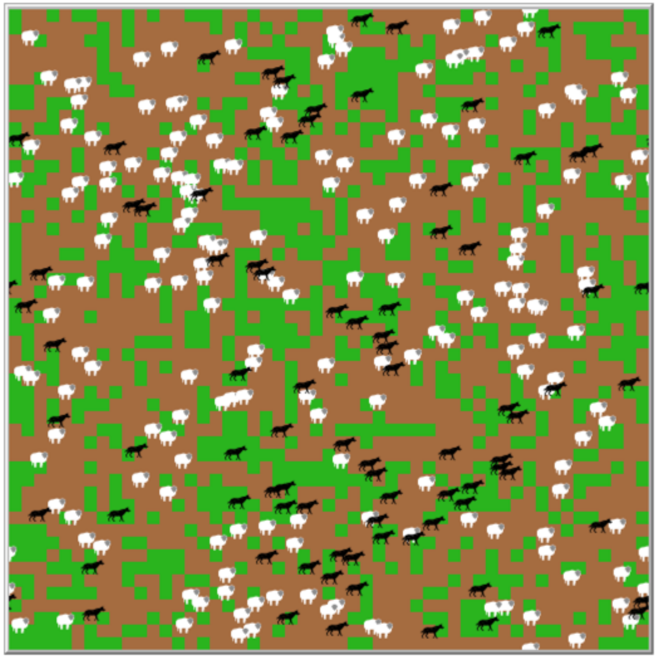
Earth:

- If it is green: do nothing
- If it is brown, wait X tick and then turn green

What do we get?

- With these new rules, we have stable fluctuations between the amount of prey and predators (and grass!).
- By limiting the sheep's resources, you increase their chances of survival (paradox of enrichment)
- Adding a level of complexity in this case led to increased stability, not more 'chaos'!





The danger of curve fitting

- In attempting to reproduce globally observed behaviour with rules, one runs the risk of arriving at a model that bears a **superficial resemblance** to the system one is trying to model, but one achieves this through '**uncorrelated mechanisms**'.
- **Agent models are less prone** to this danger than equation models, as they model systems at two levels (underlying mechanisms and global behaviour) rather than one (global behaviour).
- To avoid this risk, one should always ask oneself: **what have I missed about the behaviour of agents**? - not, simply, **how can I change my model to make it behave as I want?**

(Wilensy & Reisman, 2006)