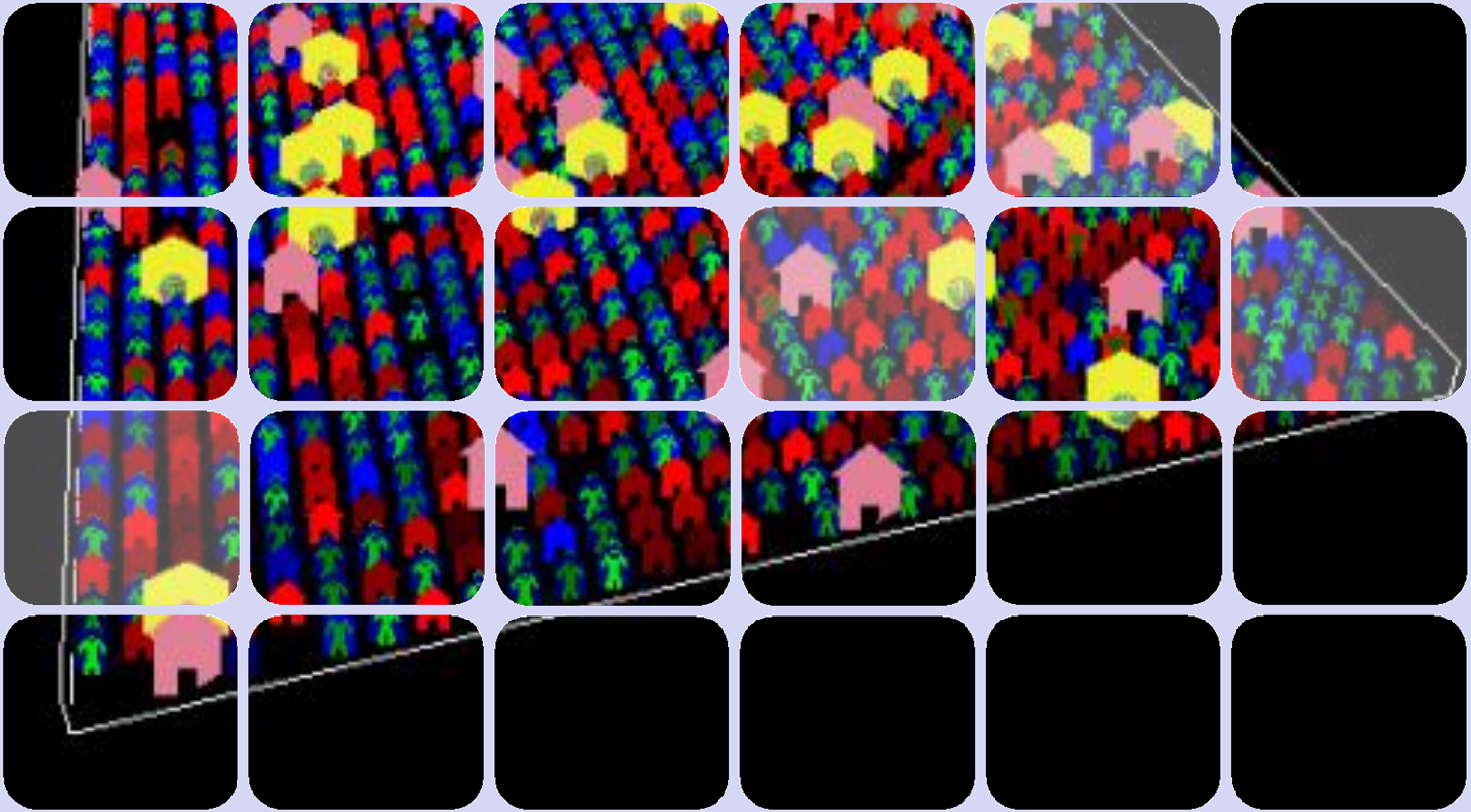
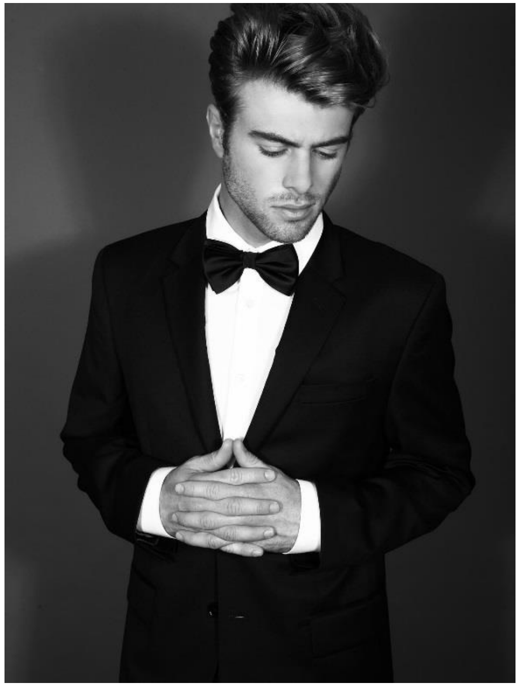
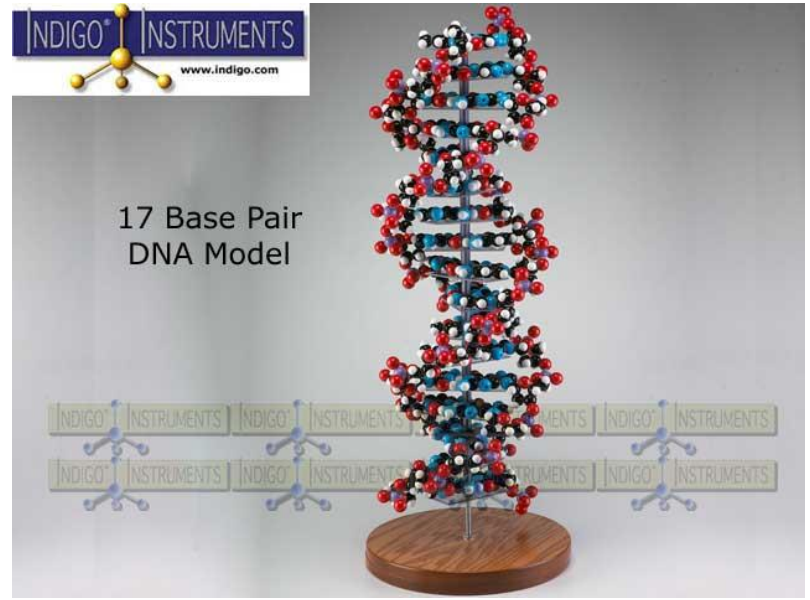
Introduction to agent-based modeling

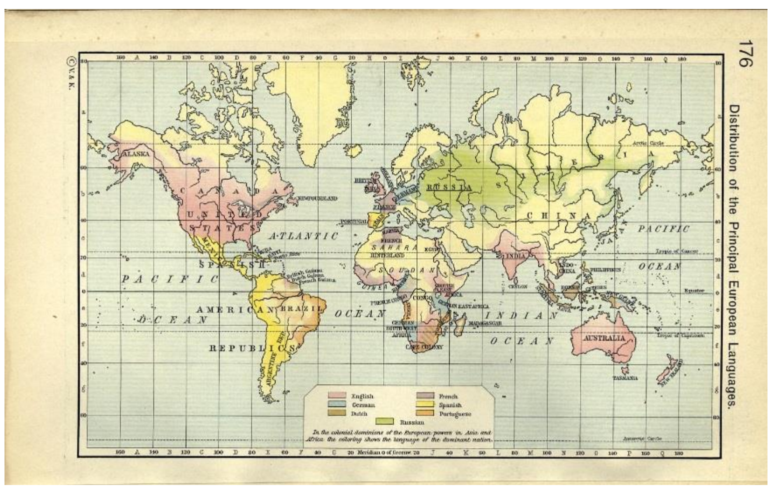


What is a model and why do we need them?

What is a model and why do we need them?





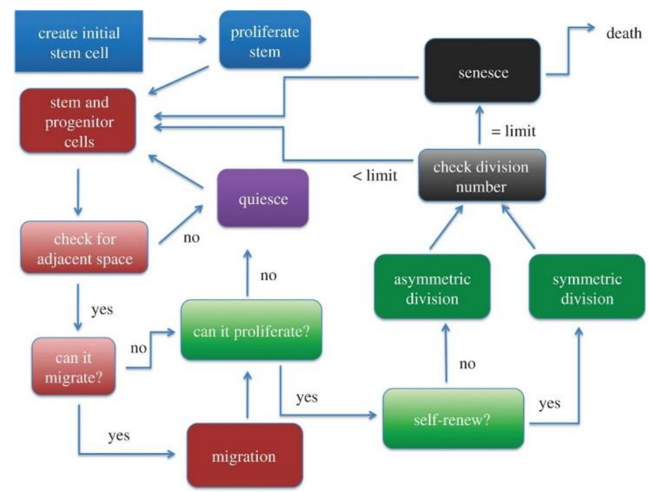


Growth rate for species 1

$$\frac{dN_1}{dt} = r_1 N_1 \left(1 - \frac{N_1}{K_1} - \frac{\alpha_{12} N_2}{K_1}\right)$$

Growth rate for species 2

$$\frac{dN_2}{dt} = r_2 N_2 \left(1 - \frac{N_2}{K_2} - \frac{\alpha_{21} N_1}{K_2}\right)$$



Models are abstract representations of the world

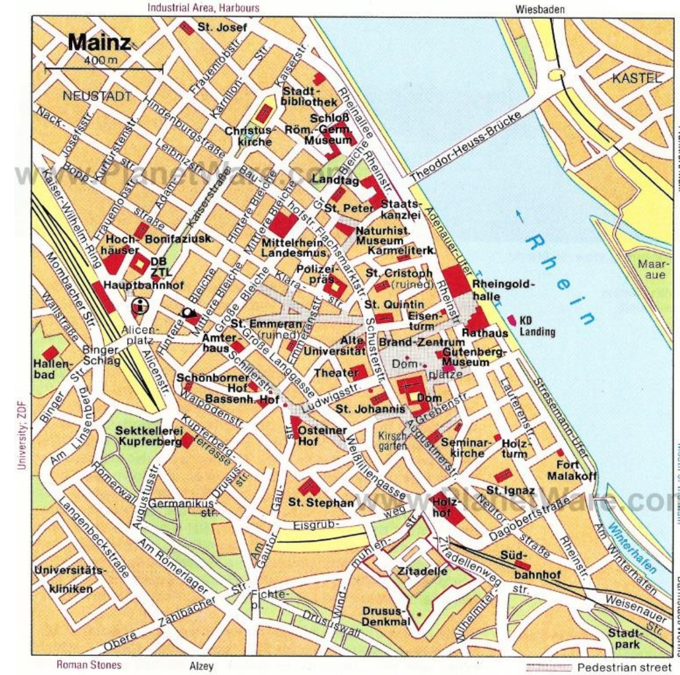
- Models miss many aspects of reality…

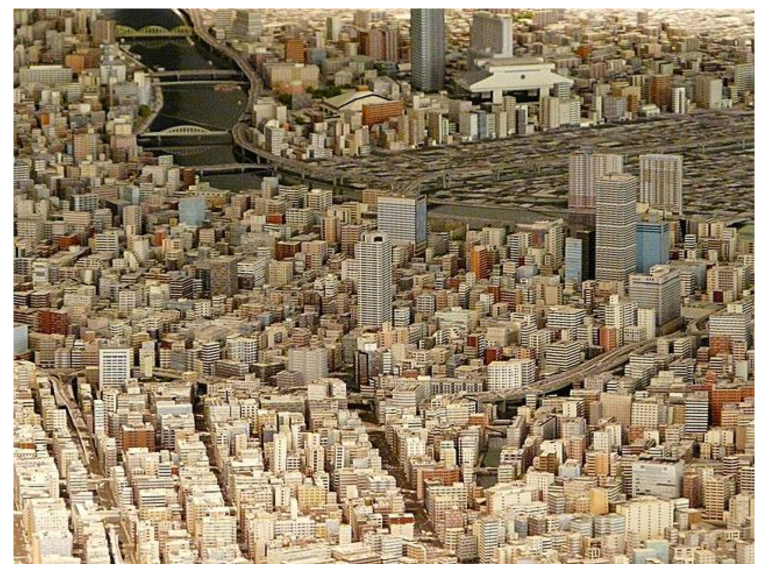
…but they capture some important aspects

- Models should contain sufficient information to make them useful

- Models can also be too realistic to be useful!

> *"All models are wrong, but some are useful"* George Box, statistician





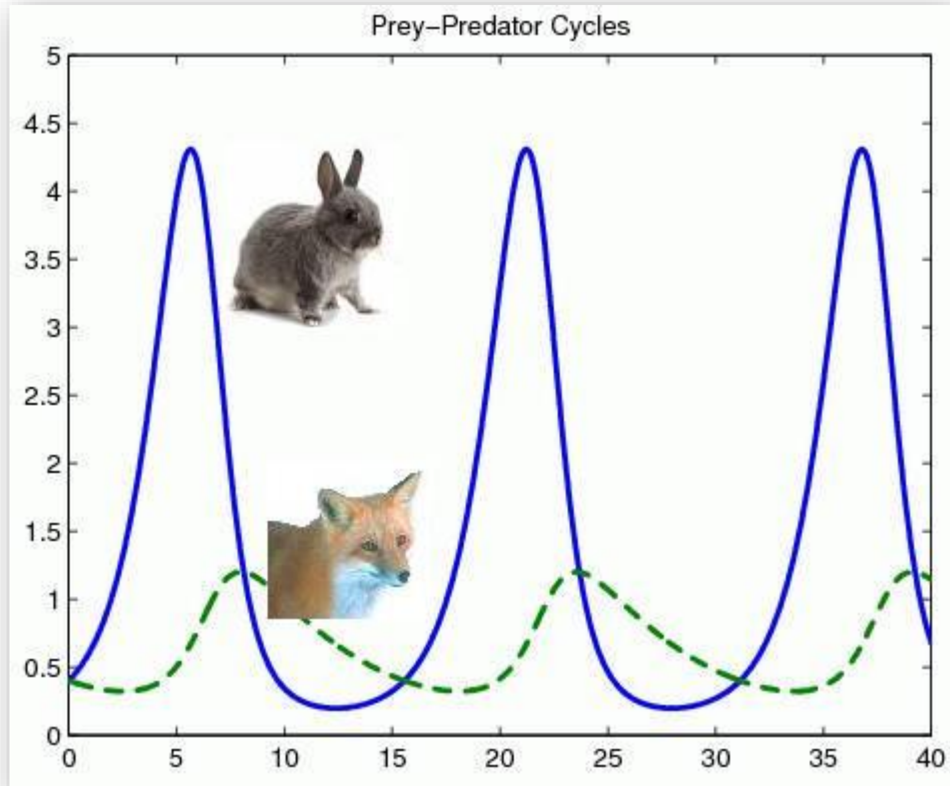
Mathematical vs. agent-based models

- Mathematical (analytical) models have a long tradition: e.g. Lotka & Volterra model

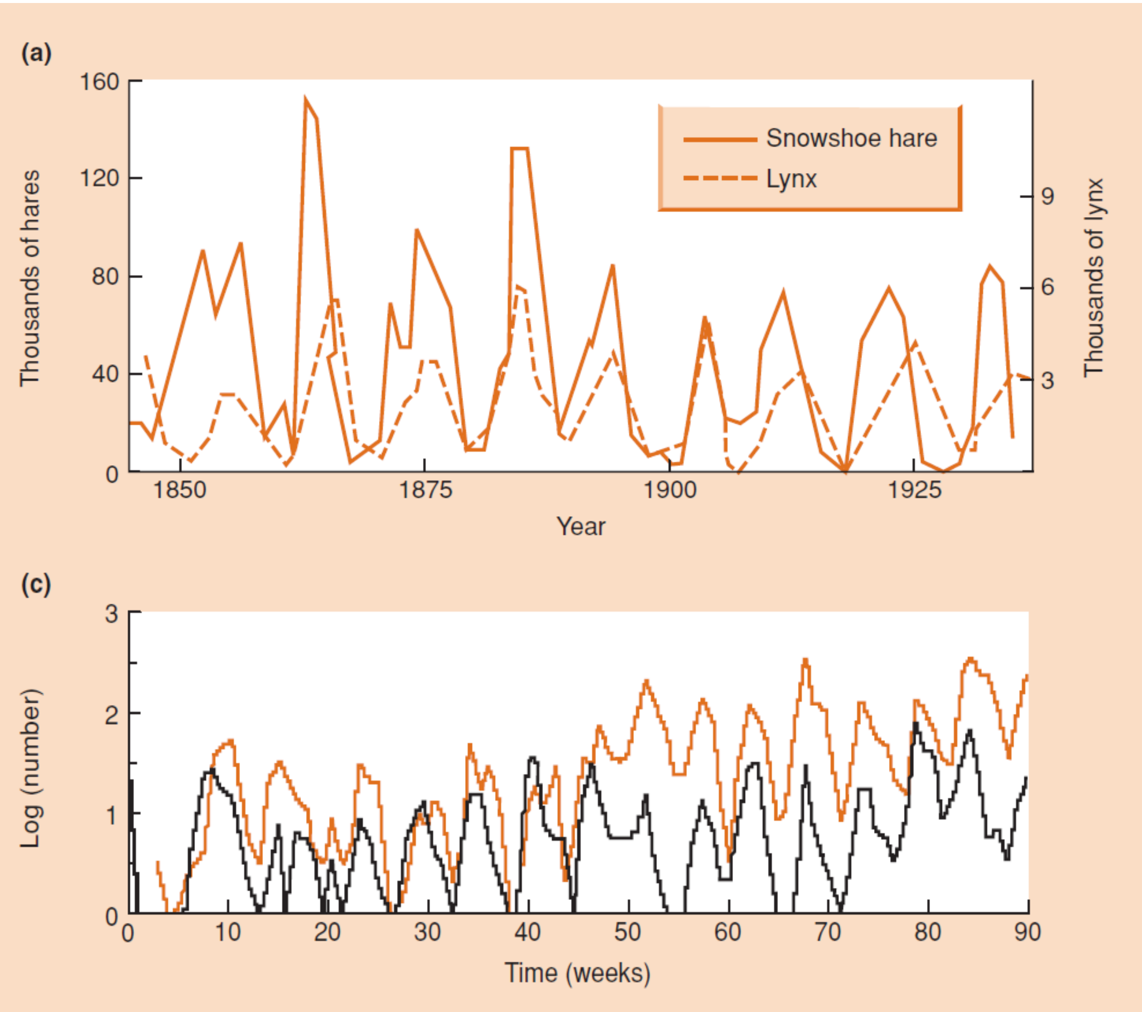
$$\frac{dN_1}{dt} = r_1 N_1 \left(1 - \frac{N_1}{K_1} - \frac{\alpha_{12} N_2}{K_1}\right)$$

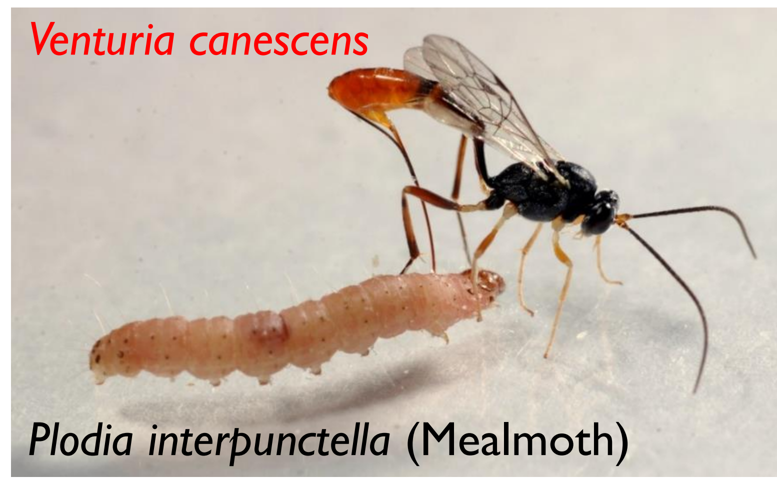
$$\frac{dN_2}{dt} = r_2 N_2 \left(1 - \frac{N_2}{K_2} - \frac{\alpha_{21} N_1}{K_2}\right)$$

N = population size
K = carrying capacity
r = rates of increase
α = competition coefficient



Do the model predictions match empirical data?





From Begon et al. 2006

Mathematical vs. agent-based models

- Agent-based simulations that have become more popular as computer use and computer power have increased

- ABM's simulate a system as a collection of autonomous decision-making entities called agents/individuals

  - An *agent* is a computational individual or object with particular properties and actions

  - An *agent* assesses its situation and makes decisions on the basis of a set of programmed rules

Agent-based models are similar to computer games

- We program an environment
- We program types of agents
- We give agents rules how to behave and interact

We run virtual experiments and collect the data

Pro's and con's of agent-based models (ABM's)

- Advantages of ABM's: its easy to study…
  - variability among individuals
  - different types of interactions
  - changes in behaviour or strategy
  - heterogeneous environments

⟹ Great to model Complex Systems

What are complex systems?

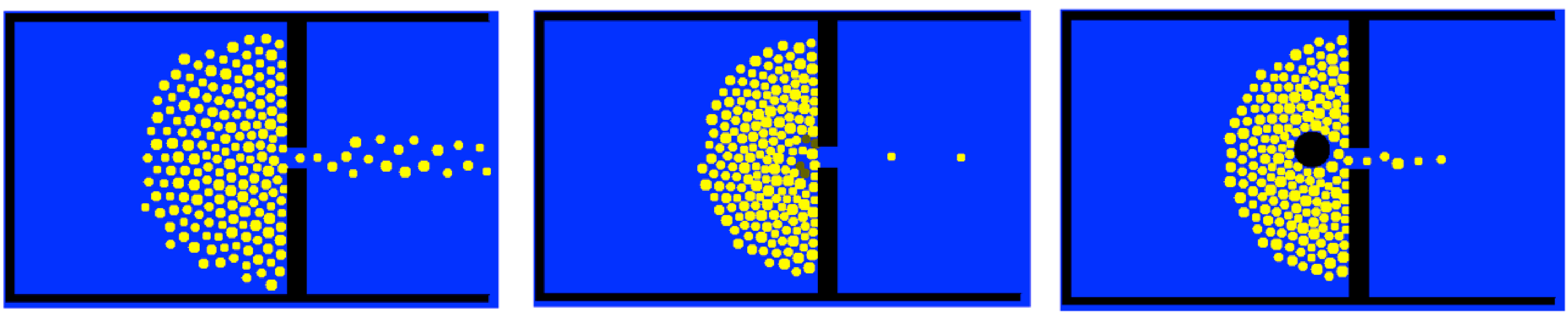
Complex systems have **emergent properties**: properties that can not be predicted by studying the individual parts

Pro's and con's of agent based models (ABM's)

- Advantages of ABM's: its easy to study…
  - variability among individuals
  - different types of local interactions
  - changes in behaviour or strategy
  - heterogeneous environments

⇒ Great to model Complex Systems

Example: simulate escape dynamics



Pro's and con's of agent based models (ABM's)

- Disadvantages of ABM's:
  - You need computers…sometimes powerful computers
  - They can be difficult to analyse and understand

⟹ They are sometimes a bit of a black box even for the programmer

Despite these differences, mathematical models and agent-based models usually provide the same conclusions!

Other advantages of ABM's

- We can model unrealistic and non-existing agents or environments

  For example, consider the following hypothesis: *Alarm calls* improve survival of offspring in *dangerous environments*

  Using ABM's we create parents that do not perform *alarm calls* (virtual mutants) and simulate offspring survival in a *virtual environment with virtual predators*

- We can test whether speculative hypotheses are theoretically consistent

Plan for today

1. Explore different examples of models in **NetLogo**

2. Program a model

3. Test the model

What is NetLogo

- A modelling language for agent-based models that…
  - …is for free (https://ccl.northwestern.edu/netlogo/)
  - …can be learned quickly
  - …sophisticated enough to do useful science:
    it is now used in many scientific publications
    (https://ccl.northwestern.edu/netlogo/references.shtml)

1. Explore different models

- Open **NetLogo → File → Models Library**

  go to **Games → Lunar Lander → setup → go**

- go to **Biology →** check out the following models (5-10 min)
  - **"Moths"**
  - **"Membrane Formation"**
  - **"Climate change" (Earth Science folder)**

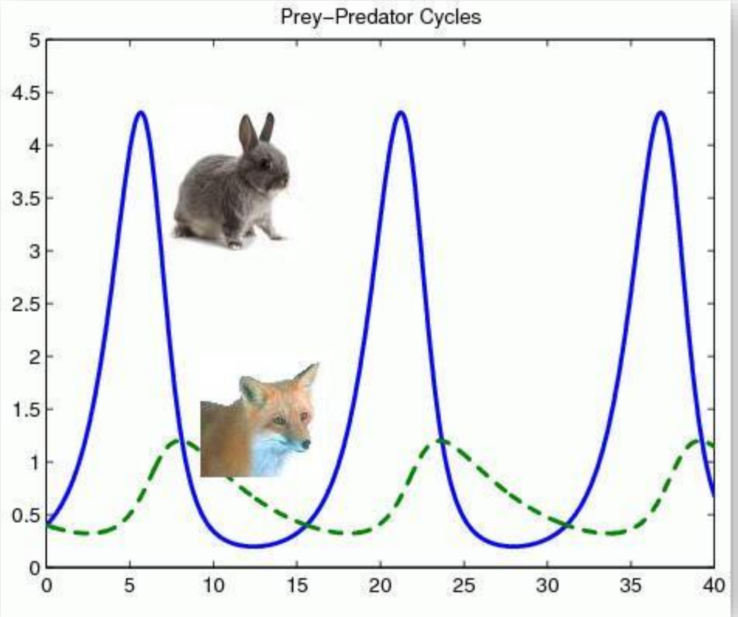
- go through the **Sample Model #1** (15 min)

NetLogo

2. Build a predator-prey model

$$\frac{dN_1}{dt} = r_1 N_1 \left(1 - \frac{N_1}{K_1} - \frac{\alpha_{12} N_2}{K_1}\right)$$

$$\frac{dN_2}{dt} = r_2 N_2 \left(1 - \frac{N_2}{K_2} - \frac{\alpha_{21} N_1}{K_2}\right)$$



2. Build a predator-prey model

- First, think about what your model needs for your purposes
  - What agents do we need?
  - What behaviours do we need?
- The values can come from real world measurements or, if no information is available, we can "*guesstimate*"

4 main sections in a code:

1. Create the agents and their characteristics ("breeds")

```
breed [agent type 1]
```

2. Setup the world

```
to setup
    …
end
```

3. Define behaviour of agents

```
to go
    …
end
```

4. Show results

```
to do-plotting
    …
end
```

- "File" → "New", then go to the "Code" section

```
;; we start by creating the agent type "sheep"
breed [sheep a-sheep] ;; always create singular and plural

;; comment your code!!!!! You might forget why you did what

;; this procedure sets up the model (the "world")
to setup
    clear-all          ;; resets all variables for a new run
    ask patches [ ;; world made of patches, colour them green
        set pcolor green
    ]
    create-sheep 100 [ ;; create the initial sheep
        setxy random-xcor random-ycor
        set color white
        set shape "sheep"
    ]
    reset-ticks        ;; makes sure time starts at 0
end ;; "end" always indicates the end of a command
```

- Check the code for errors using the "Check" button

- Create button “*setup*” & “*go*” in interface (go: choose “forever”)

```
;; this procedure starts actions, place after "to setup"
to go
    ask sheep [ ;;"ask" asks agents to do something
        wiggle ;; first turn a little bit in each direction
        move   ;; then step forward
    ]
    tick       ;; after agents performed actions, tick +1
end

;; sheep procedure, the sheep changes its heading
to wiggle
        right random 90 ;; picks a random value from 0-90
        left random 90
end

to move
        forward 1  ;; 1 refers to 1 patch length
end
```

- Check if the model works
- Now our turtles need further properties

```
;; living & moving cost energy, we give sheep energy
sheep-own [ energy ] ;; add after sheep are defined

;; add energy & create a number-of-sheep slider

to setup
    clear-all
    ask patches [ ;; colour the world green
        set pcolor green
    ]
    create-sheep number-of-sheep [        ;; set to 100
        setxy random-xcor random-ycor
        set color white
        set shape "sheep"
        set energy 100
    ]
    reset-ticks
end
```

- Create slider called “number-of-sheep” (300 as maximum)

```
;; moving costs energy

to move
      forward 1
      set energy energy - 1 ;; take away a unit of
                            ;;energy with every time step
end
```

- Check if the model and the slider work

```
;; to make the "energy" value more meaningful, sheep die
;; if energy is 0

to go
   ask sheep [
      wiggle
      move
      check-if-dead ;; checks to see if sheep dies
   ]
   tick
end
```

```
;; sheep procedure: if energy is low, sheep dies, add after
;; move procedure

to check-if-dead
      if energy < 0 [
         die
      ]
end
```

- Run the model and observe. What do you see?

```
;; make model stop when all sheep are dead
to go
   if not any? sheep [ stop ]
   ask sheep [
      wiggle
      move
      check-if-dead
   ]
   tick
end
```

- It would be useful to know how many sheep there are!

```
;; plot the number of sheep
to go
    if not any? sheep [ stop ]
    ask sheep [
        wiggle
        move
        check-if-dead
    ]
    tick
    my-update-plots ;; plot the population counts
end
```

- Now we need to define the "*my-update-plots*" procedure

```
to my-update-plots
    plot count sheep
end
```

- Create a plot and name it as you like. Rename the "Pen update commands" to "plot count sheep"

- Create a slider for the energetic costs

```
to move
    forward 1
    set energy energy - movement-cost  ;; set 1, max 5
end
```

- Now we need to give sheep the ability to eat grass and gain energy

```
patches-own [ grass-amount ] ;; patches have amounts of grass
```

- We need to set up the grass amount and colour the patches to indicate how much grass there is

```
to setup
   clear-all
   ask patches [
      set grass-amount random-float 10.0 ;; amount of food
      set pcolor scale-color green grass-amount 0 10
      ;; the brighter the green, the more grass
   ]
```

- Check the model, set it up several times and see colour

- Make sheep *eat* the grass

```
to go
    if not any? Sheep [ stop ]
    ask sheep [
        wiggle
        move
        check-if-dead
        eat             ;; new procedure
    ]
    tick
    my-update-plots
end

;; sheep procedure, sheep eat grass, add before plotting
to eat
    if(grass-amount >= 1) [   ;; 1st check if there is grass
    set energy energy + 1     ;; increase sheep energy
    set grass-amount grass-amount - 1 ;; decrease the grass
    set pcolor scale-color green grass-amount 0 10
    ]
end
```

- Check the model. What happens?
- The model behaviour is still not satisfactory. All sheep die after they have eaten the grass
- What could be changed?
- To make it more interesting we add a procedure the make the grass *regrow*

```
to go
    if not any? Sheep [ stop ]
    ask sheep [
        wiggle
        move
        check-if-dead
        eat
    ]
    regrow-grass       ;; the grass grows back
    tick
    my-update-plots
end

;; regrow the grass procedure
to regrow-grass
    ask patches [
        set grass-amount grass-amount + 0.1 ;; add 0.1 every
                                             ;;time step
        if grass-amount > 10 [
            set grass-amount 10   ;; limit grass amount to 10
        ]
        set pcolor scale-color green grass-amount 0 10
    ]
end
```

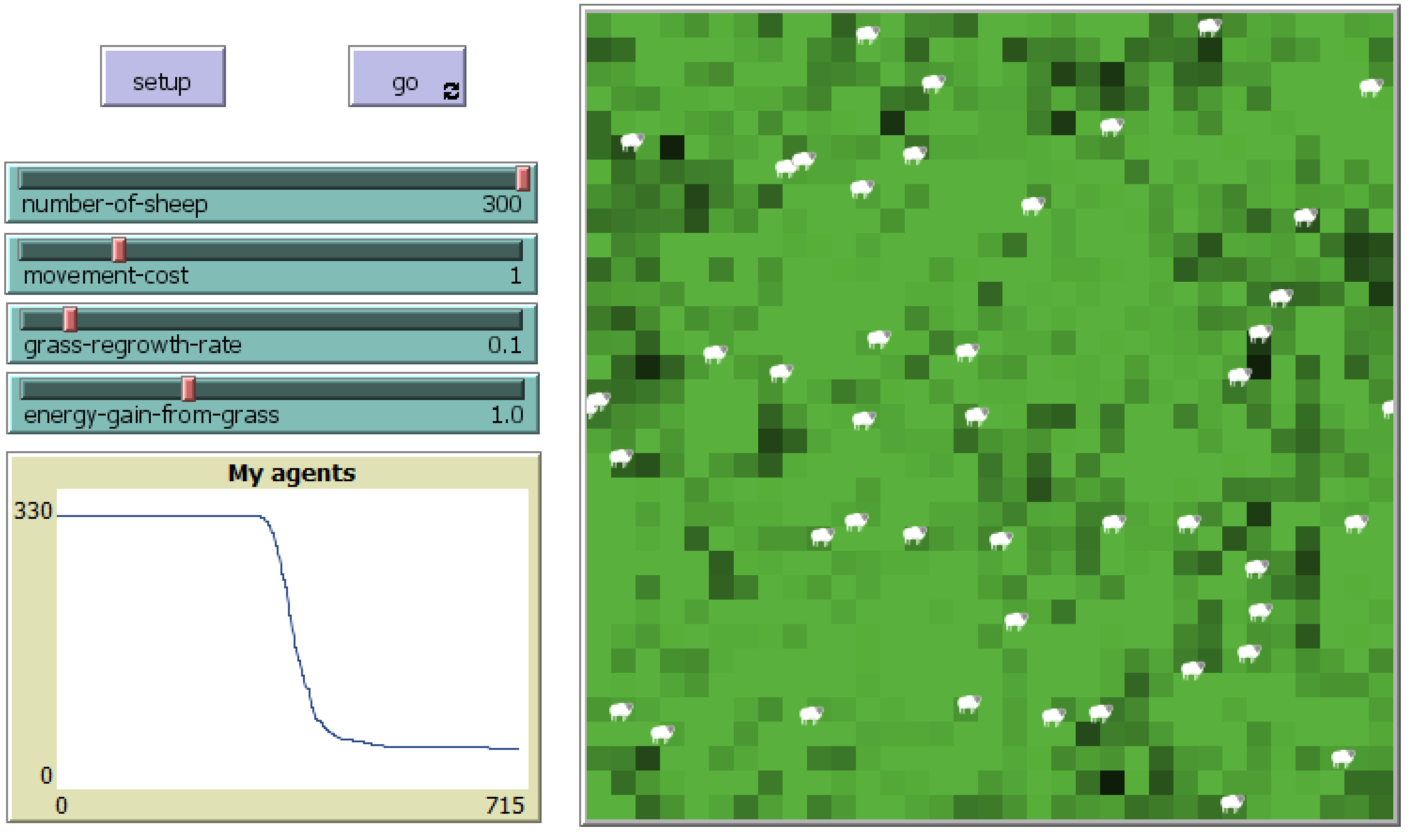
- Run the model a few times. Run it with 100 sheep or with 300 sheep. What do you observe?
- It seems that the grass-growth rate is an important parameter, so we want to be able to change it easily with a slider

```
;; regrow the grass procedure
to regrow-grass
    ask patches [
        set grass-amount grass-amount + grass-regrowth-rate
    ;set to 0.1, max 1.0
            if grass-amount > 10 [
            set grass-amount 10
        ]
        set pcolor scale-color green grass-amount 0 10
    ]
end
```

- Try changing the *growth rate* and see what happens
- Now we also add a *energy-gain* slider

```
;; sheep procedure, sheep eat grass
to eat
    if(grass-amount >= energy-gain-from-grass) [    ;set 1
    set energy energy + energy-gain-from-grass
    set grass-amount grass-amount - energy-gain-from-grass
    set pcolor scale-color green grass-amount 0 10
    ]
end
```

- Try the following configuration. Can you reproduce the mass-starvation event after around 170 time steps?



- How can the sheep population recover?

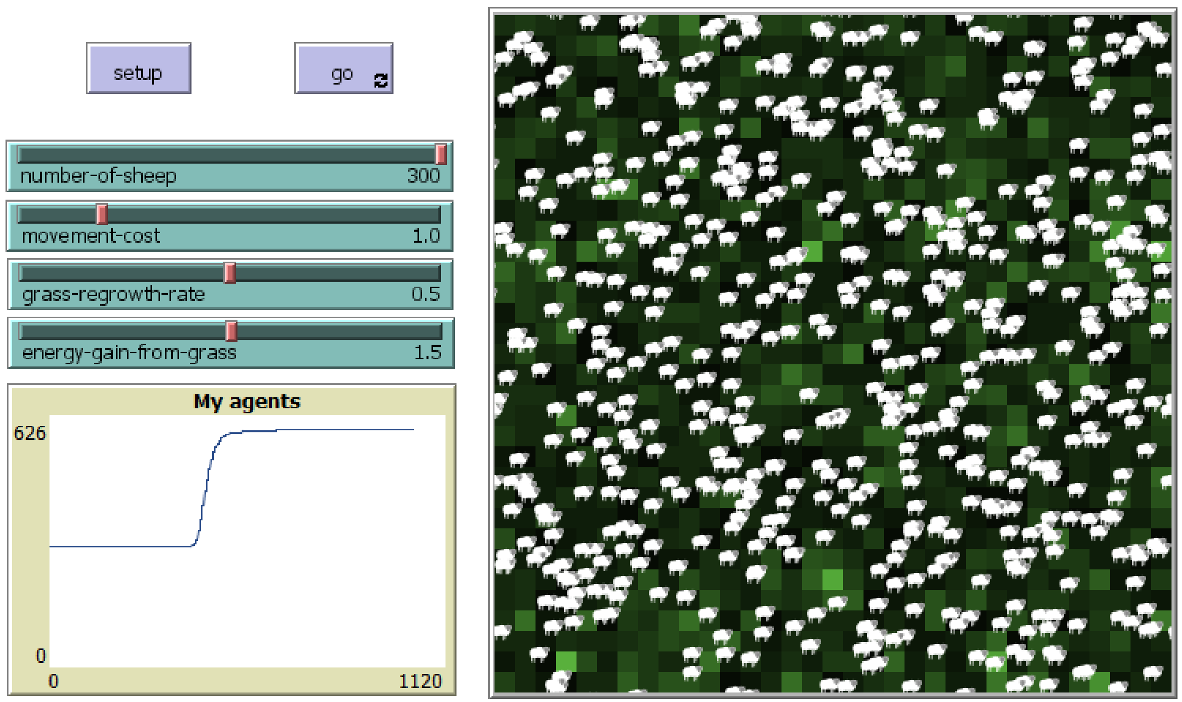
- For our purposes, simple asexual reproduction will be enough (we don’t need to model sexual reproduction, pregnancy, parental care ect.)
- Second assumption: sheep will reproduce when they reached a certain energy level

```
to go
    if not any? sheep [ stop ]
    ask sheep [
        wiggle
        move
        check-if-dead
        eat
        reproduce
    ]
    regrow-grass
    tick
    my-update-plots
end
```

- Now we create a reproduce-procedure

```
;; check to see if this sheep has enough energy to reproduce
to reproduce
    if energy > 200 [
        set energy energy - 100 ;; reproduction requires energy
        hatch 1 [ set energy 100 ] ;; create sheep with energy
    ]
end
```

- Now what happens with very productive grass?



- Our sheep agent is done, now we need wolves!
- Programming the wolves will be easier because we already have most of the code, we just need to add some lines

```
;; we add the wolf "breed"
breed [sheep a-sheep]
breed [wolves wolf]

;; replace the sheep-own property with the more generic
;; turtles-own property. It means that it applies to all agents
sheep-own [ energy ]
turtles-own [ energy ]
```

- Now we need to add the wolves & a wolf slider:

```
;; to add wolves, copy the sheep-code and modify it
to setup
    clear-all
    ask patches [
        set grass-amount random-float 10.0
        recolor-grass
    ]
    create-sheep number-of-sheep [
        setxy random-xcor random-ycor
        set color white
        set shape "sheep"
        set energy 100
    ]
    create-wolves number-of-wolves [ ;; create 50 wolves, max 300
        setxy random-xcor random-ycor
        set color red
        set shape "wolf"
        set size 1.5       ;; make them a bit bigger
        set energy 100
    ]
    reset-ticks
end
```

- Check the model!

- Now we need to add their behaviour
- Because many behaviours are the same for sheep and wolves (wiggling, moving, dying, reproducing), we replace "*sheep*" in the *go* procedure with the general "*turtle*"

```
to go
    if not any? turtles [ stop ]
    ask turtles [
        wiggle
        move
        check-if-dead
        eat
        reproduce
    ]
    regrow-grass
    tick
    my-update-plots
end
```

- However, the eating behaviour of wolves should be different! We need to modify the "*eat*" procedure

- First, we rename our old "*eat*" procedure "*eat-grass*", then we add our new, more general "*eat*" procedure

```
;; sheep eat grass, wolves eat sheep
to eat
    ifelse breed = sheep [
        eat-grass
    ]
    [
        eat-sheep
    ]
end
```

- Now we must define our "*eat-sheep*" procedure

```
;; wolves eat sheep
to eat-sheep
    if any? sheep-here [  ;; checks if a sheep is on the same
                        ;; patch
        let target one-of sheep-here ;; selects a random sheep on
                                    ;; my patch
        ask target [ die ] ;; eat the selected sheep
        set energy energy + energy-gain-from-sheep
    ]
end
```

- Before running the model, don't forget to add a slider "energy-gain-from-sheep" (start with 5.0 as value, max. 10)
- Now the model has all the agents, behaviours, and interactions we want. However, our graph doesn't show all the information we need. We need to modify the "*my-update-plots*" procedure

```
to my-update-plots
    set-current-plot-pen "sheep"
    plot count sheep

    set-current-plot-pen "wolves"
    plot count wolves

end
```

- Edit the plot → add 2 pen's → provide pen name's "sheep", "wolves

- You finished the model! Now can you get populations to fluctuate?

- Which factors help to create stable fluctuations?

- Ideally we would want to measure or estimate the parameters from real data. At the moment, the values are very unrealistic

- But the model shows that there are conditions, when these populations fluctuate

- What else could you add?