

# Agent Based Modelling for Stochastic Decision Making

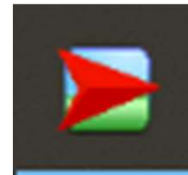
George van Voorn (Applied Mathematics & Statistics, Wageningen UR)

*August 29, 2025*

Two 'commercials':

SiLiCo (Simulating Life's Complexity)  
[George.vanvoorn@wur.nl](mailto:George.vanvoorn@wur.nl)

Innovation committee Mathematics  
[innovatie@platformwiskunde.nl](mailto:innovatie@platformwiskunde.nl)



**NETLOGO installed?**

# A little background...

- Bachelor Biology
- MSc Mathematical biology / biological modelling
- Double PhD developing global bifurcation analysis methods applied to mathematical ecology
- Model quality auditor at WUR (*models as decision support*)
- Associate professor in resilience modelling
- One core topic: agent-based modelling



# Content

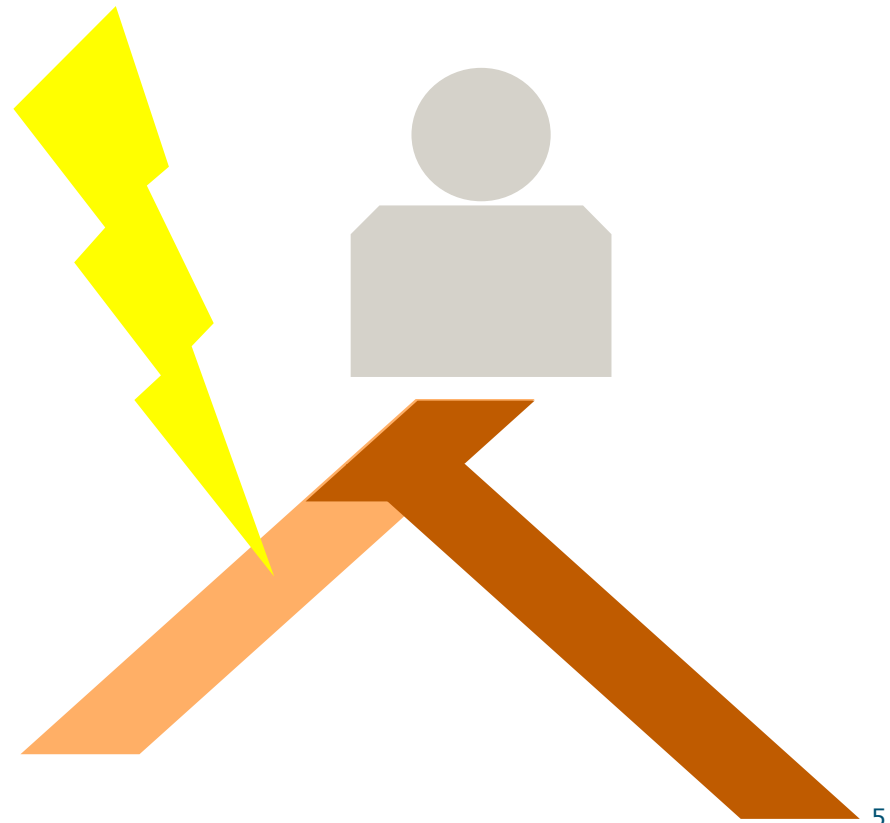
- Three parts:
- Part 1: motivation on multi-agent modelling and basic NetLogo
- Part 2: coding agent interactions in NetLogo
- Part 3: human behavioural theory

# Goal today:

- Anne introduced a uniform (generic) mathematical framework for stochastic decisions: applies to one agent (“a travelling salesman”)
- Agents Based Modelling (ABM) to simulate (semi-)stochastic decision making for **multiple interacting agents**
- Why?
  - Simulate agents (people, software), their decision making, and effects of their interactions
- How?
  - Implemented in NetLogo (didactic tool)

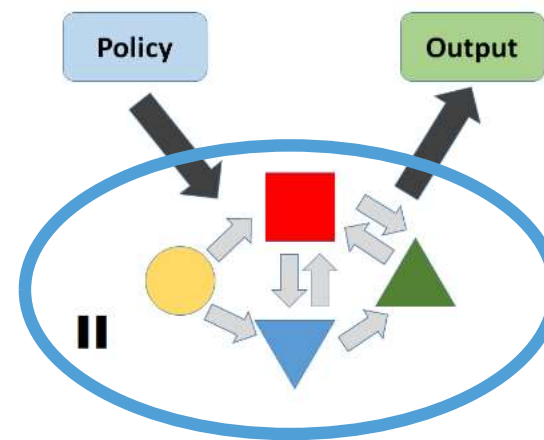
# Agency: Capacity to do things differently

- An agent = an **autonomous** decision maker and action taker (animal, human, robot, company, government,...) who exerts sufficient influence on their surroundings to be quantitatively considered (in our case, in a simulation) / relevant and who may **adapt** based on inputs from other agents and their environment



# Decisions in a multi-actor setting

- In a multi-actor setting, decisions lead to emergent patterns
- With unpredictable or undesired results
- There may not even be an optimum (Pareto) or it depends on the agent what the optimum for them is



Many of these components are interacting agents

# Decisions... Decisions...

- Idealized as a flow diagram (next slide)
- Many 'decisions' are
  - effectively 'semi' stochastic: random effects but rule-based ("biased die"), unless you have theory and data
  - subconscious or bounded in rationality

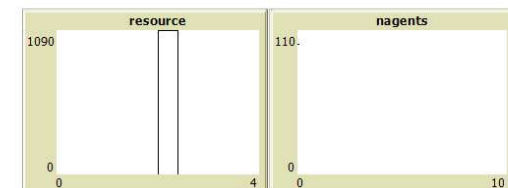
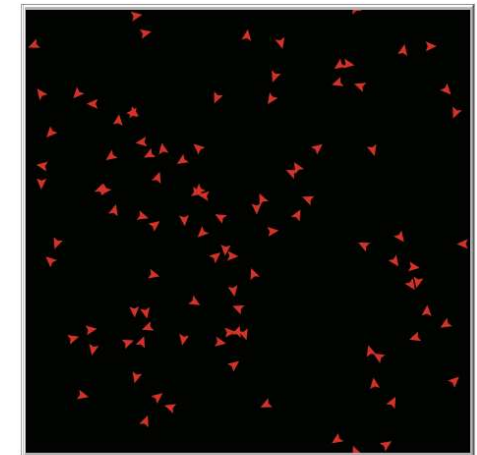
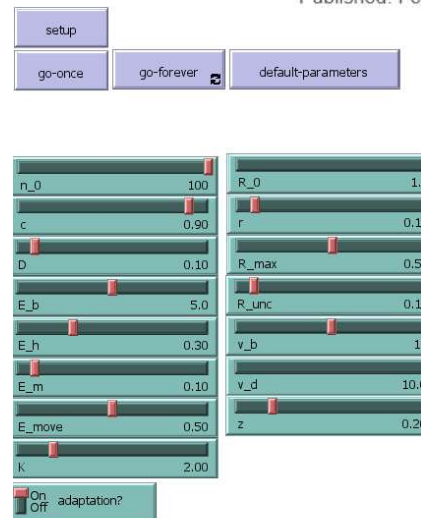
**PLOS ONE**

## Resilience through adaptation

Guus A. ten Broeke , George A. K. van Voorn, Arend Ligtenberg, Jaap Molenaar

Published: February 14, 2017 • <https://doi.org/10.1371/journal.pone.0171833>

 OPEN ACCESS  PEER-REVIEWED



# Resilience example model description & interface

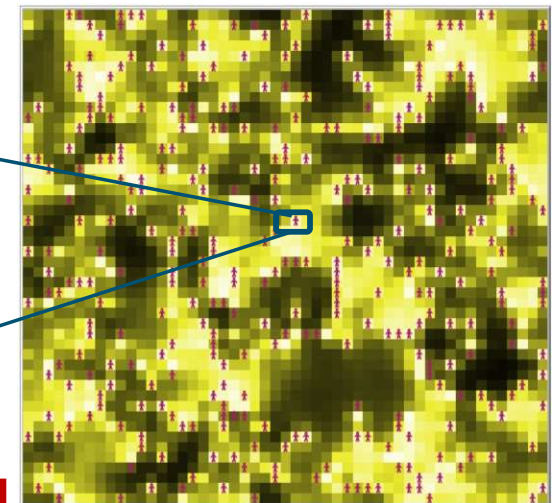
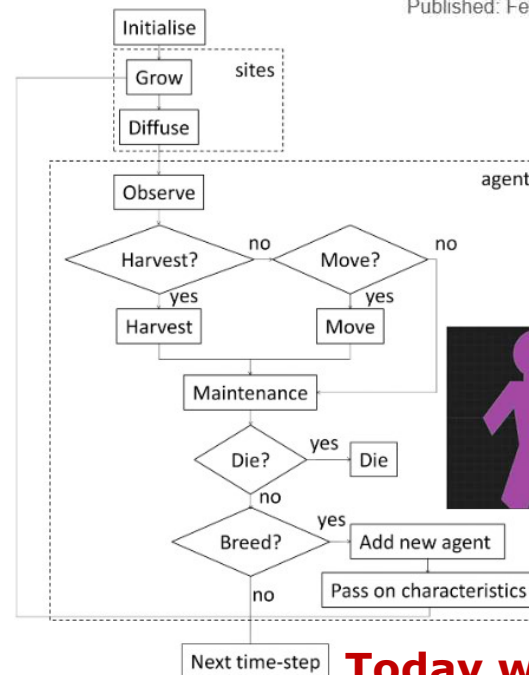
- Logistically growing and diffusing resource
- Consumers: Harvest decision depends on:
  - presence resource
  - closeness of others
  - internal energy state
- Procreate (inheritance) and die (depends on internal energy)

PLOS ONE

## Resilience through adaptation

Guus A. ten Broeke, George A. K. van Voorn, Arend Ligtenberg, Jaap Molenaar

Published: February 14, 2017 • <https://doi.org/10.1371/journal.pone.0171833>

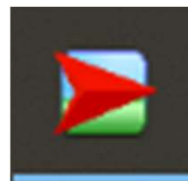


**Today we will  
recreate a lot of  
this "machinery"**



# ABM: Suitable tool for multi-agent simulations

- ***Start up your laptops!***
- We will build this up from the ground with toy examples in NetLogo
- To simulate:
  - Individual decision making in agents
  - Interactions between agents and their environment
- We will also look at some social theory-based decision modelling



**OPEN UP NETLOGO!**

# NetLogo model library

NetLogo

File Edit Tools Zoom Tabs Help

New Ctrl+N

Open... Ctrl+O

Models Library Ctrl+M

Recent Files

Save Ctrl+S

Save As... Ctrl+Shift+S

Upload To Modeling Commons

Save As NetLogo Web...

Export

Import

Print... Ctrl+P

Quit Ctrl+Q

normal speed

view updates

ticks:

contin...

Settings...

Models Library

Sample Models

- Art
- Biology
- Chemistry & Physics
- Computer Science
- Earth Science
  - Climate Change
  - Continental Divide
  - Erosion
  - Fire
  - Grand Canyon
  - Percolation
  - River Meanders
- (unverified)
- Games
- Mathematics
- Networks
- Philosophy
- Psychology
- Social Science
- System Dynamics

Curricular Models

- BEAGLE Evolution
- Connected Chemistry
- CT-STEM
- epiDEM
- GasLab
- GenEvo
- Lattice Land
- MaterialSim
- Mind the Gap
- ModelSim
- NIELS
- PNoM
- ProLab
- Urban Suite

Code Examples

HubNet Activities

IABM Textbook

Go to User Community Models web page

Open Cancel

Fire

This project simulates the spread of a fire through a forest. It shows that the fire's chance of reaching the right edge of the forest depends critically on the density of trees. This is an example of a common feature of complex systems, the presence of a non-linear threshold or critical parameter.

# Tab Interface

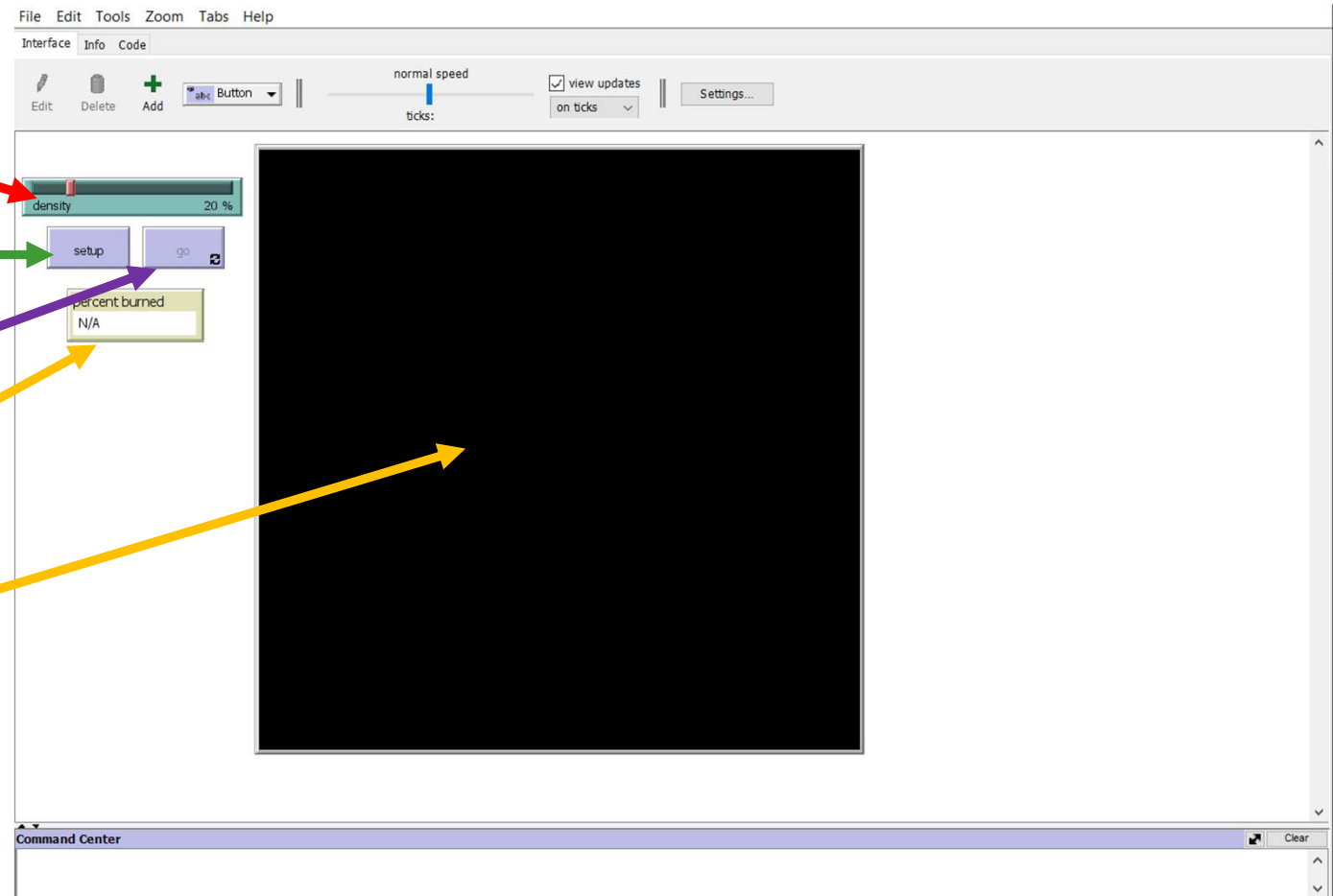
Sliders: initial conditions, scenarios, parameters

Initialization

We can only go after we have initialized

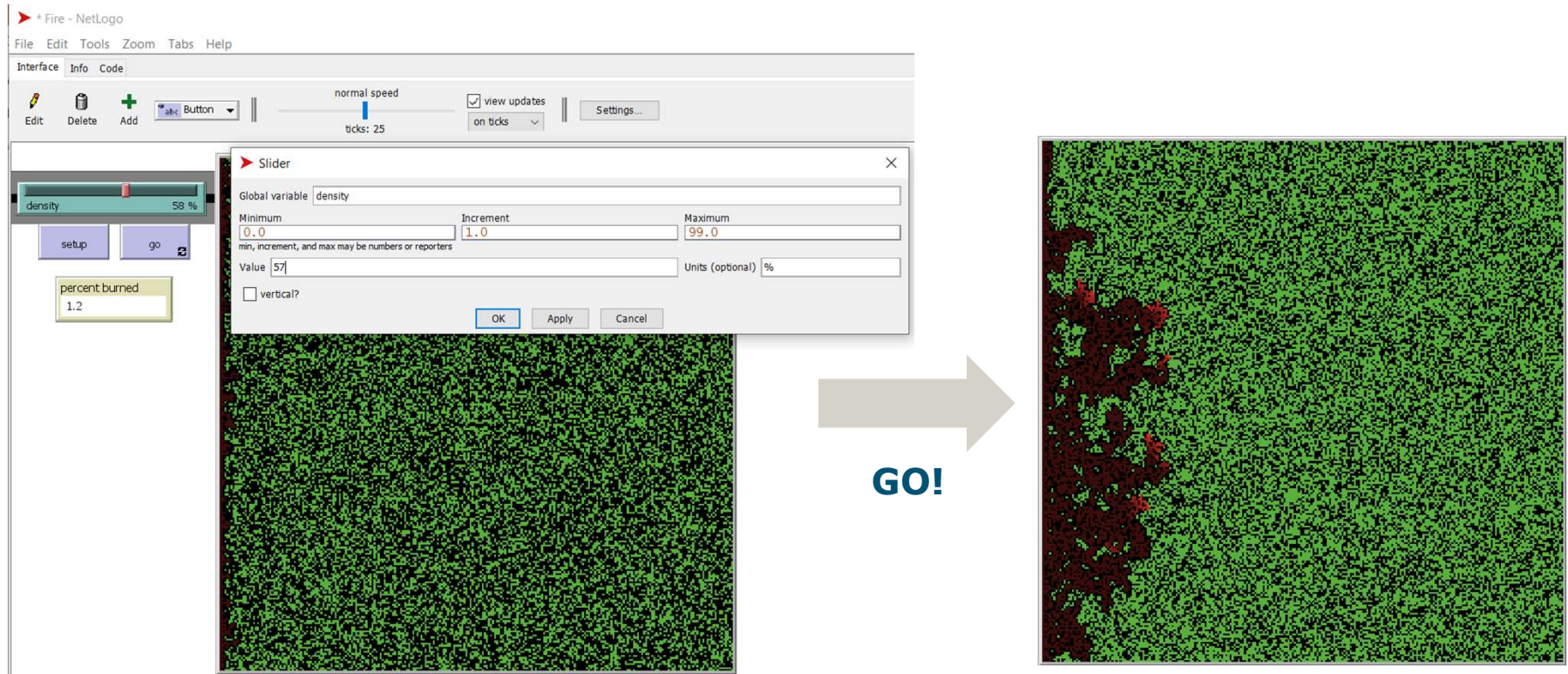
Output (numbers)

Output ("the world", i.e., spatiotemporal)



# Example run

**Right-click on slider / Edit, set value to 20, then Apply / OK. Then Setup**

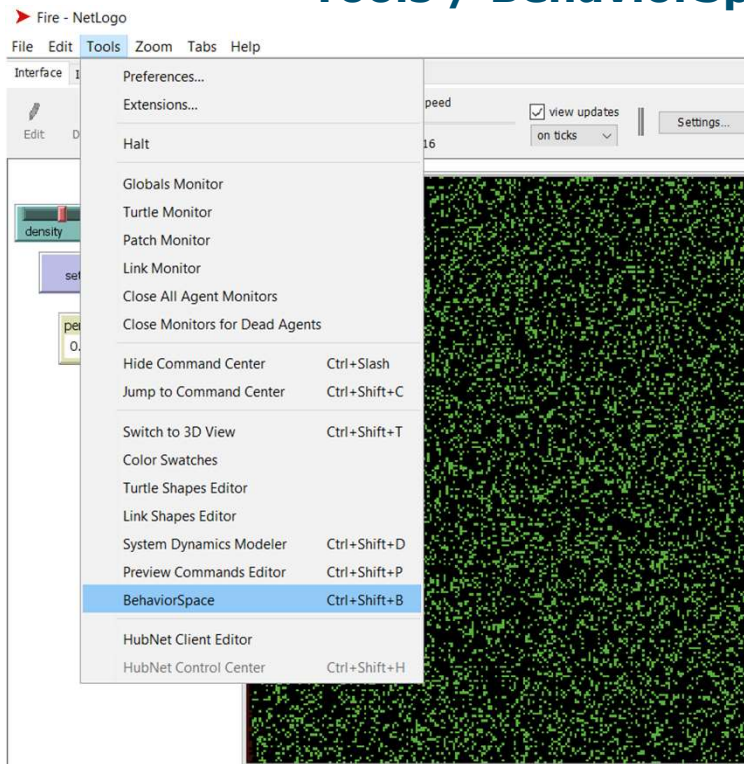


You can stop the simulation by pressing Go again

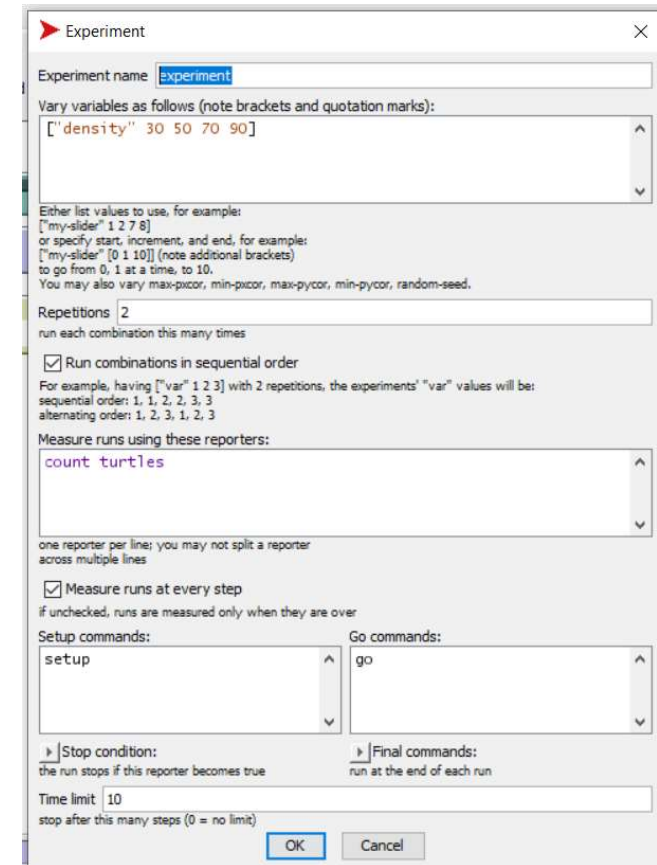


# Sensitivity analysis with Behaviour Space

## Tools / BehaviorSpace ; new

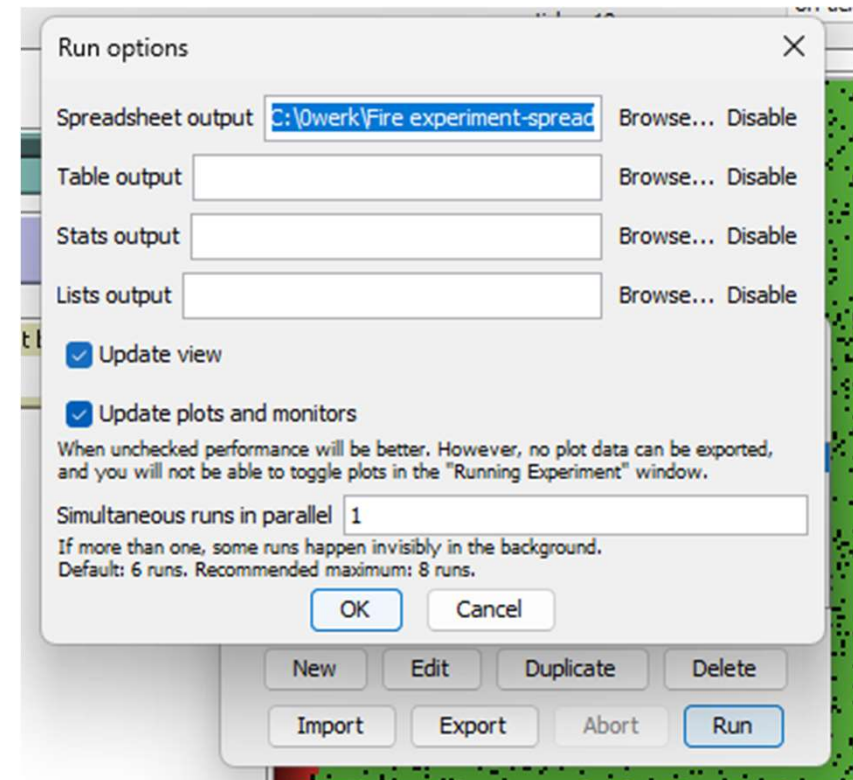


- Let's test first: "density" at 30, 50, 70, 90
- Two replicates
- Stop after 10 ticks



# Sensitivity analysis with Behaviour Space (2)

- Press 'Run'
- To get Excel file in tab delimited:
  - select column A
  - Data / Text to columns
  - Delimited
  - Tab and Comma
  - Next, Next, Finish



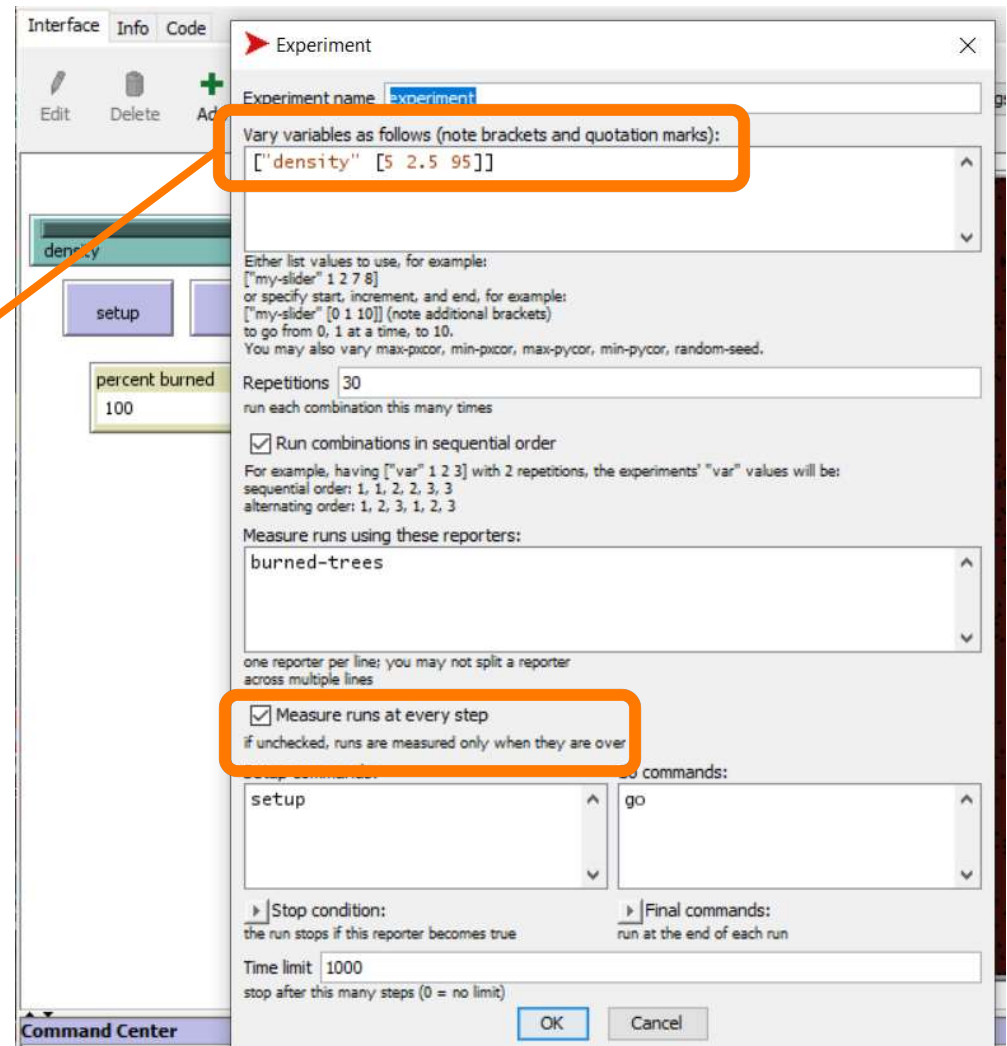
# Output test

- Csv-file
- Density
- Output per tick

1	BehaviorSpace results (NetLogo 6.1.1)								
2	Fire.nlogo								
3	experiment								
4	01/24/2022 13:52:24:545 +0100								
5	min-pxcor	max-pxcor	min-pycor	max-pycor					
6	-125	125	-125	125					
7	[run numb	1	2	3	4	5	6	7	8
8	density	30	30	50	50	70	70	90	90
9	[reporter]	count turtl	count turtl	count turtl	count turtl	count turtl	count turtl	count turtl	count turtles
10	[final]	408	407	654	619	1416	1562	2339	2302
11	[min]	251	251	251	251	251	251	251	251
12	[max]	408	407	654	619	1416	1562	2339	2302
13	[mean]	375.2727	379.4545	500.2727	506.6364	851.1818	926.6364	1295.182	1284.273
14	[steps]	10	10	10	10	10	10	10	10
15									
16	[all-run dat	count turtl	count turtl	count turtl	count turtl	count turtl	count turtl	count turtl	count turtles
17		251	251	251	251	251	251	251	251
18		336	339	362	385	418	428	468	475
19		360	366	411	441	528	557	661	674
20		377	387	460	487	636	684	866	876
21		388	397	495	515	742	804	1081	1079
22		395	402	524	539	849	923	1297	1292
23		399	405	547	560	964	1048	1506	1493
24		402	406	572	579	1071	1186	1717	1694
25		405	407	602	591	1186	1309	1926	1894
26		407	407	625	606	1302	1441	2135	2097
27		408	407	654	619	1416	1562	2339	2302

# Full OFAT

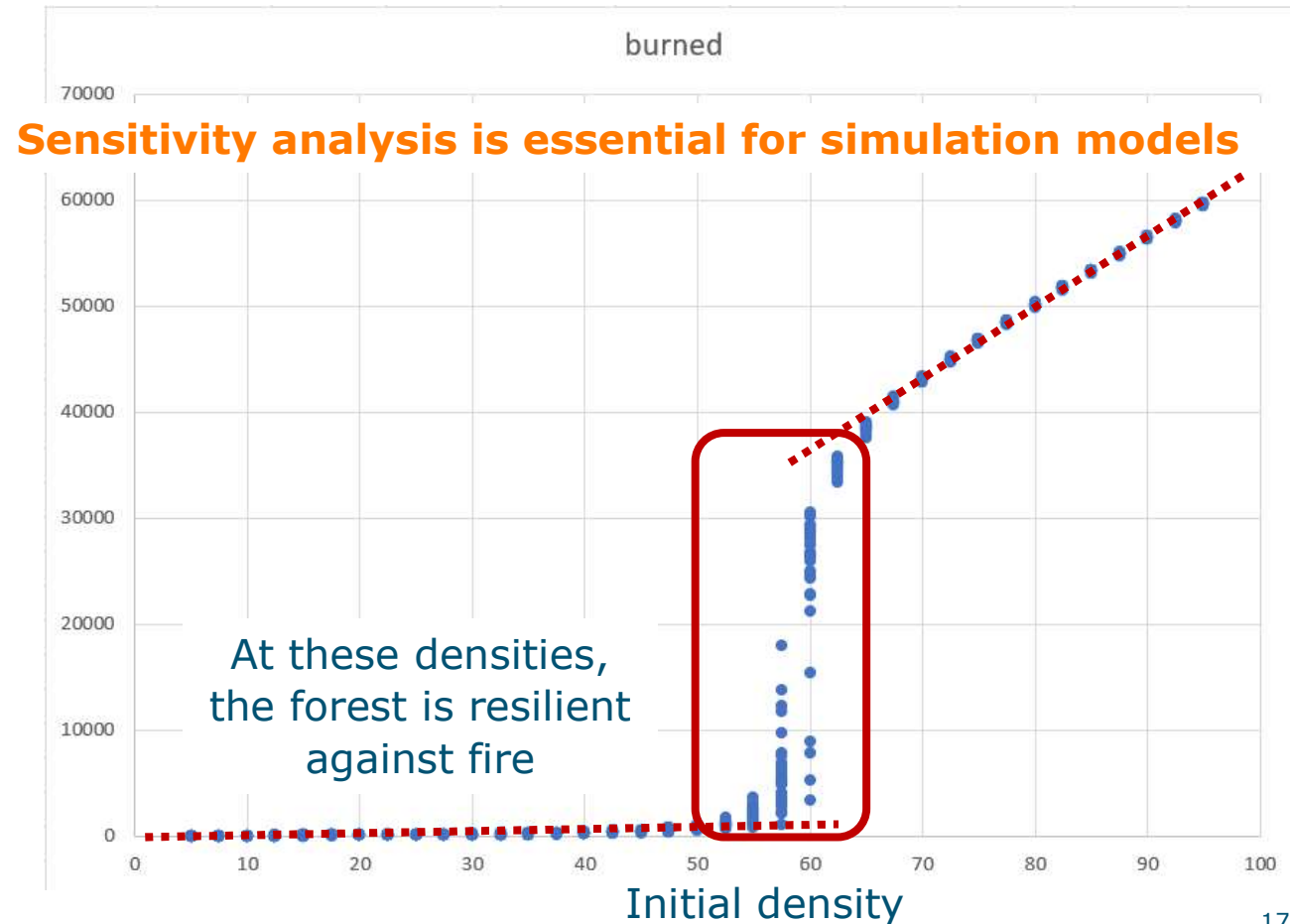
- From 5% to 95%
- Increment of 2.5%
- Notation: ["..." [5 2.5 95]]
- 'burned-trees'
- 30 replicates
- 1000 ticks
- 1110 runs (APS = last column)





# Plot of output of Fire model

- What's happening?
- A critical threshold in initial density for fires to expand
- ABM:
  - Spatiotemporal
  - Interactions
  - Noisy





# Reporting an ABM: Tab Info

Interface

Info

Code

Find...

Edit

## WHAT IS IT?

This project simulates the spread of a fire through a forest. It shows that the fire's chance of reaching the right edge of the forest depends critically on the density of trees. This is an example of a common feature of complex systems, the presence of a non-linear threshold or critical parameter.

## HOW IT WORKS

The fire starts on the left edge of the forest, and spreads to neighboring trees. The fire spreads in four directions: north, east, south, and west.

The model assumes there is no wind. So, the fire must have trees along its path in order to advance. That is, the fire cannot skip over an unwooded area (patch), so such a patch blocks the fire's motion in that direction.

## HOW TO USE IT

Click the **SETUP** button to set up the trees (green) and fire (red on the left-hand side).

Click the **GO** button to start the simulation.

The **DENSITY** slider controls the density of trees in the forest. (Note: Changes in the **DENSITY** slider do not take effect until the next **SETUP**.)

## THINGS TO NOTICE

When you run the model, how much of the forest burns. If you run it again with the same settings, do the same trees burn? How similar is the burn from run to run?

Each turtle that represents a piece of the fire is born and then dies without ever moving. If the fire is made of turtles but no turtles are moving, what does it mean to say that the fire moves?

# Documenting an ABM

- ODD (Overview, Design concepts, and Details; Grimm et al.)
- Standard protocol for ABMs published in JASSS, EMS, etc.

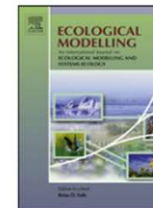
Ecological Modelling 221 (2010) 2760–2768



Contents lists available at ScienceDirect

Ecological Modelling

journal homepage: [www.elsevier.com/locate/ecolmodel](http://www.elsevier.com/locate/ecolmodel)

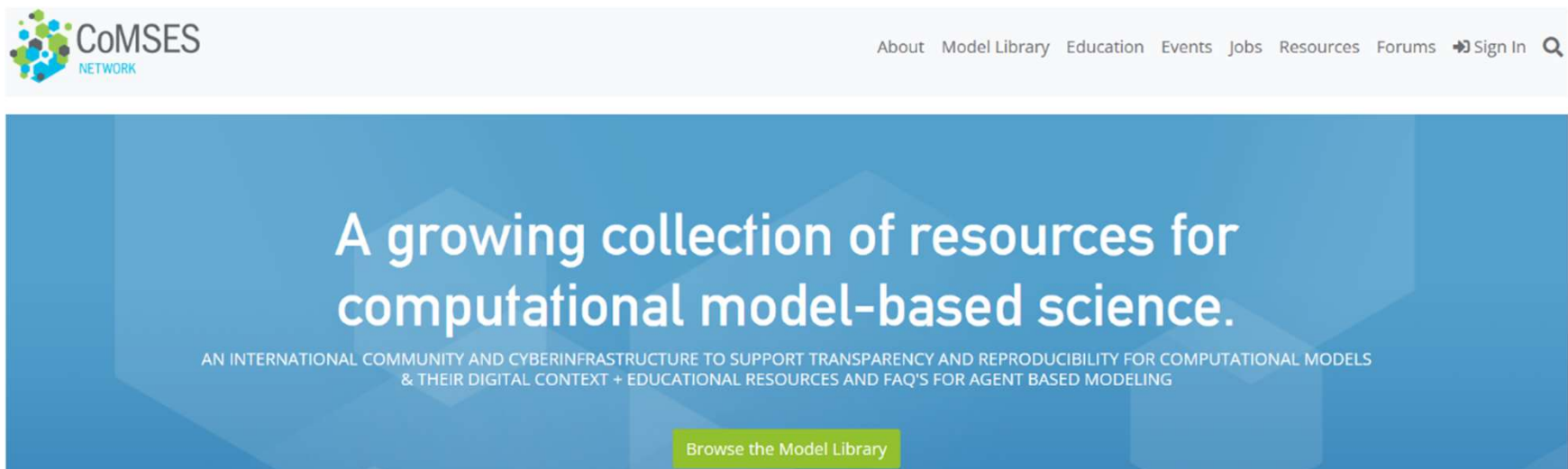


## The ODD protocol: A review and first update

Volker Grimm<sup>a,\*</sup>, Uta Berger<sup>b</sup>, Donald L. DeAngelis<sup>c</sup>, J. Gary Polhill<sup>d</sup>, Jarl Giske<sup>e</sup>, Steven F. Railsback<sup>f,g</sup>

# Archiving an ABM

- JASSS and Comses / OpenABM are common outlets for the publication and archiving of ABMs <https://www.comses.net/>



# Tab Code

Globals: generic parameters,  
inputs, outputs we monitor, etc.

Agent types:  
patches, turtles

Loops for main  
modules

Nested  
submodule loop

```
* Fire - NetLogo
File Edit Tools Zoom Tabs Help
Interface Info Code
Find... Check Procedures Indent automatically Code Tab in separate window

globals [
  initial-trees ;; how many trees (green patches) we started with
  burned-trees  ;; how many have burned so far
]

breed [fires fire] ;; bright red turtles -- the leading edge of the fire
breed [embers ember] ;; turtles gradually fading from red to near black

to setup
  clear-all
  set-default-shape turtles "square"
  ;; make some green trees
  ask patches with [(random-float 100) < density]
  [ set pcolor green ]
  ;; make a column of burning trees
  ask patches with [pxcor = min-pxcor]
  [ ignite ]
  ;; set tree counts
  set initial-trees count patches with [pcolor = green]
  set burned-trees 0
  reset-ticks
end

to go
  if not any? turtles ;; either fires or embers
  [ stop ]
  ask fires
  [ ask neighbors4 with [pcolor = green]
    [ ignite ]
    set breed embers ]
  fade-embers
  tick
end

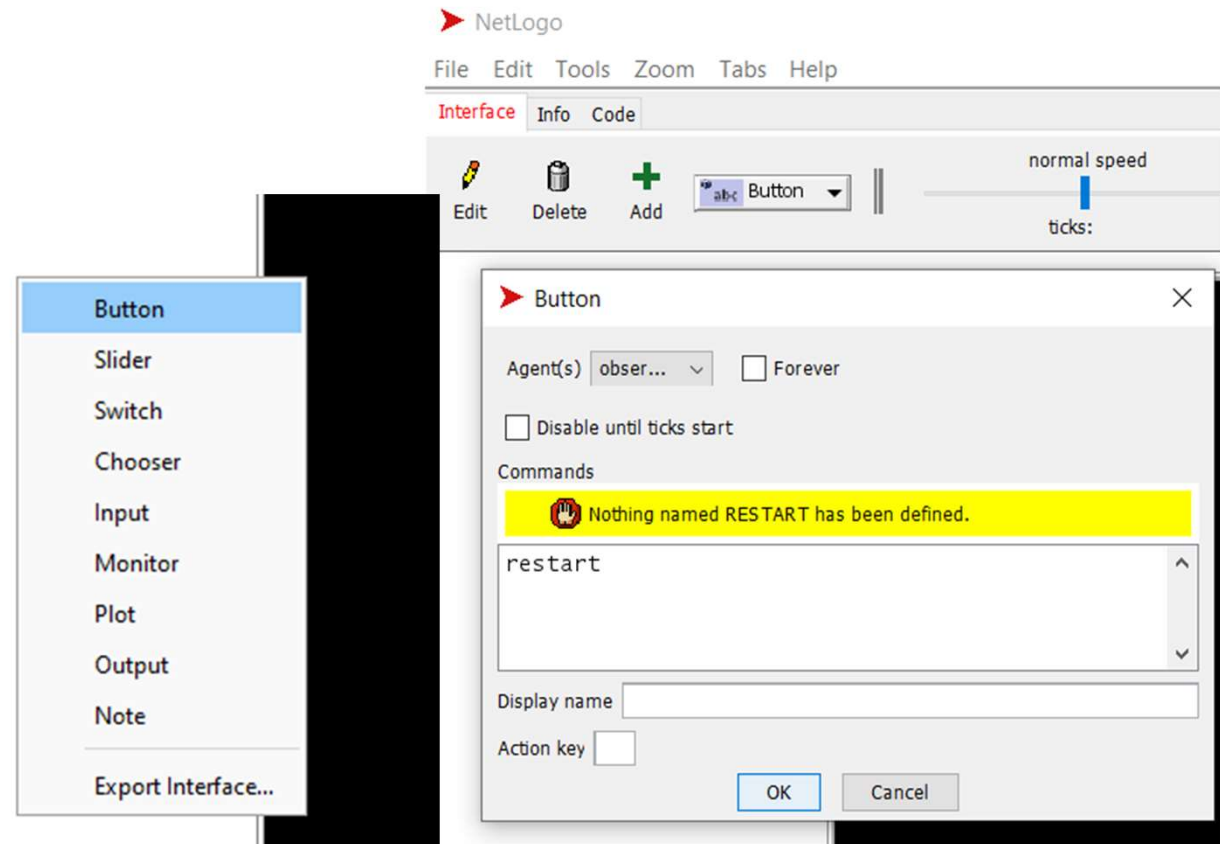
;; creates the fire turtles
to ignite ;; patch procedure
  sprout-fires 1
  [ set color red ]
  set pcolor black
  set burned-trees burned-trees + 1
end
```

# Let us start with some programming

## Create New

Tab **Interface**, create **Button**  
Click mouse somewhere  
Edit commands: restart

You get an error

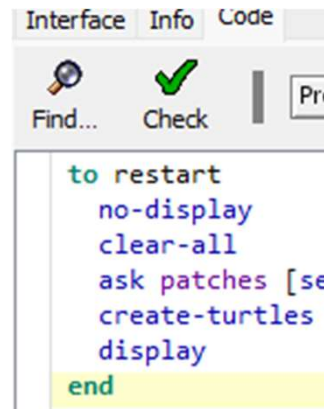


# Programming

Tab **Code**, enter:

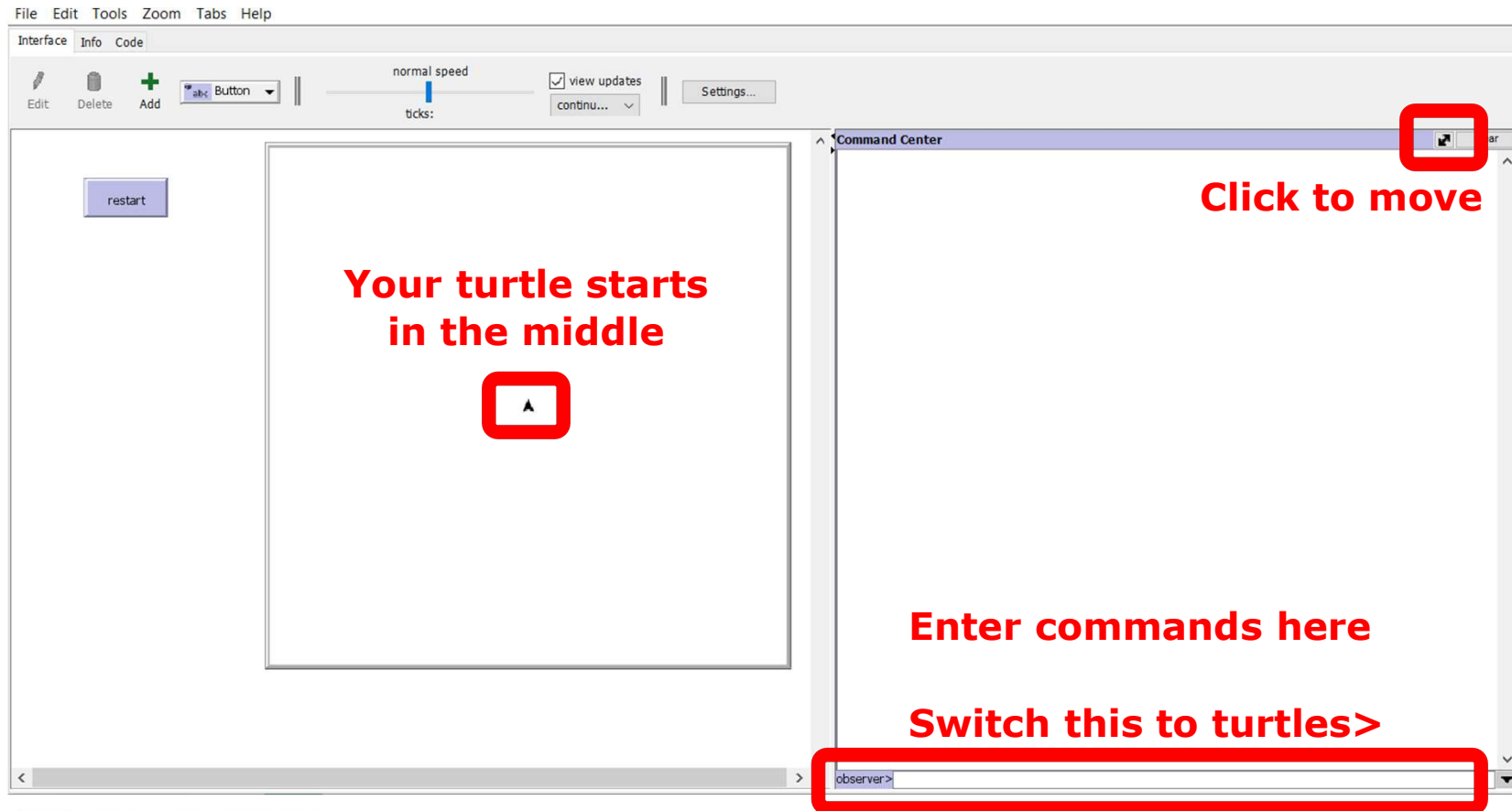
```
to restart
  no-display
  clear-all
  ask patches [set pcolor white]
  create-turtles 1 [set heading 0 set color black pen-down]
  display
end
```

**As good practice, click the green  
'check' button**



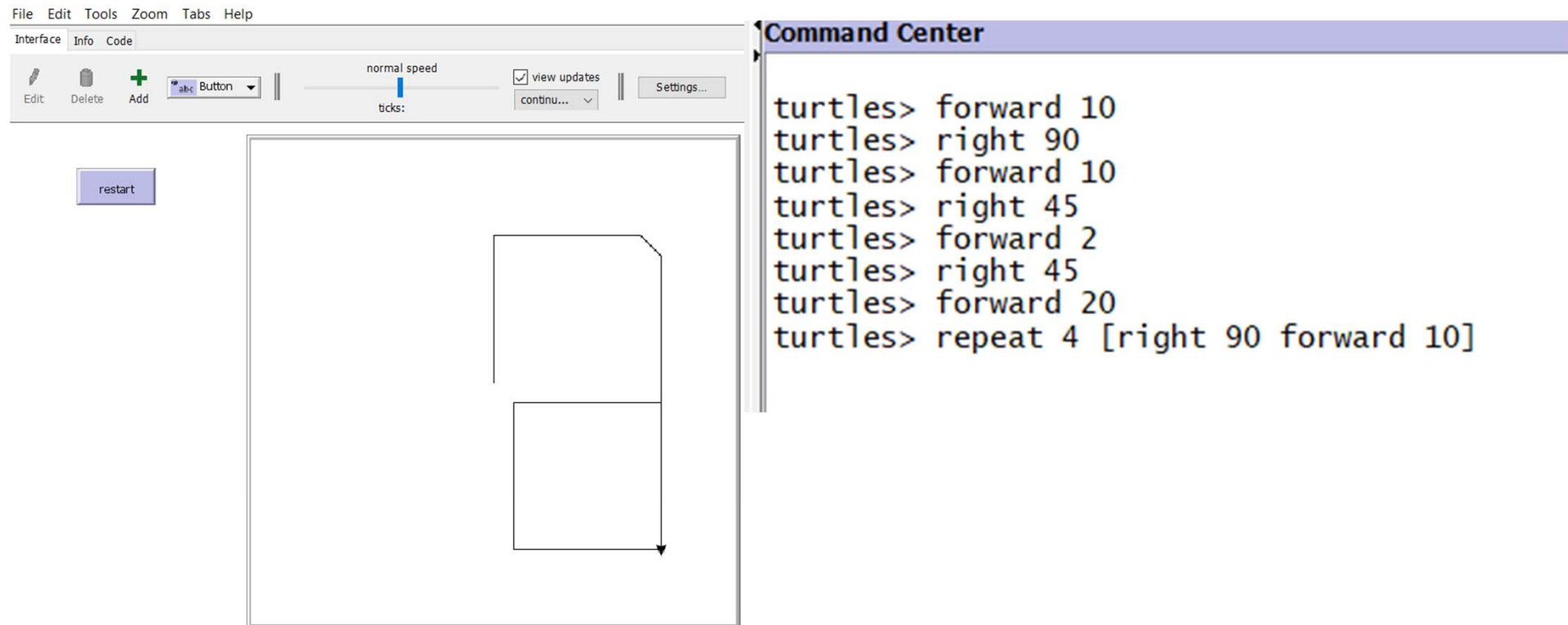
**After saving, check  
'Interface' to see if the  
error has resolved**

# The result of the programming





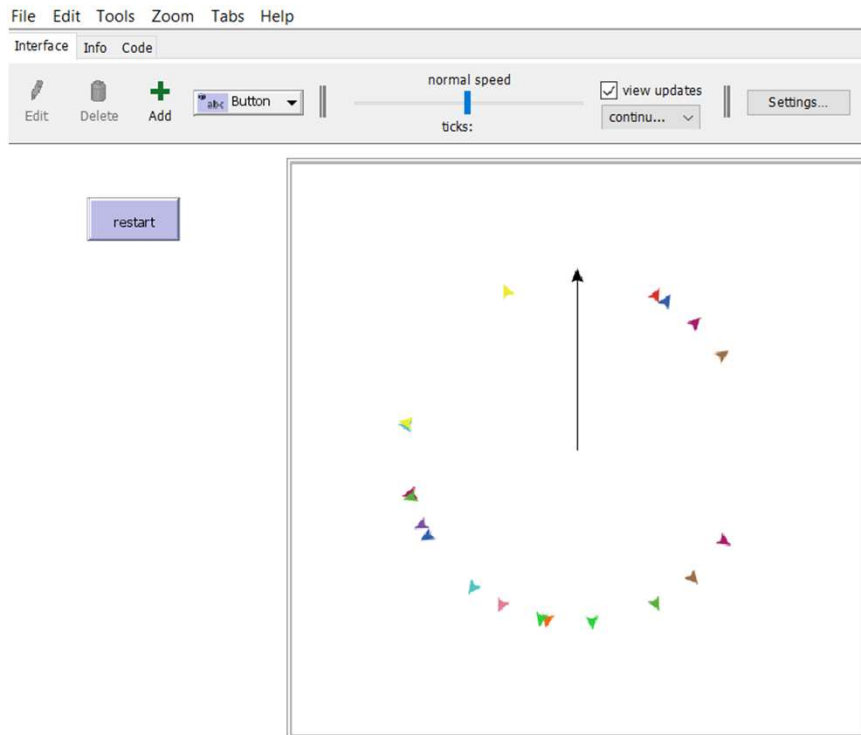
# Basic turtle commands



The screenshot displays a turtle graphics application window. The window has a menu bar (File, Edit, Tools, Zoom, Tabs, Help) and a toolbar with buttons for Edit, Delete, Add, and a Button dropdown. A speed slider is set to 'normal speed' and a 'view updates' checkbox is checked. A 'restart' button is located on the left. The main canvas shows a drawing of a square with a notch on its right side. To the right of the canvas is a 'Command Center' panel with a list of commands:

```
turtles> forward 10
turtles> right 90
turtles> forward 10
turtles> right 45
turtles> forward 2
turtles> right 45
turtles> forward 20
turtles> repeat 4 [right 90 forward 10]
```

# Basic observer commands



**Switch back to observer>**

**Command Center**

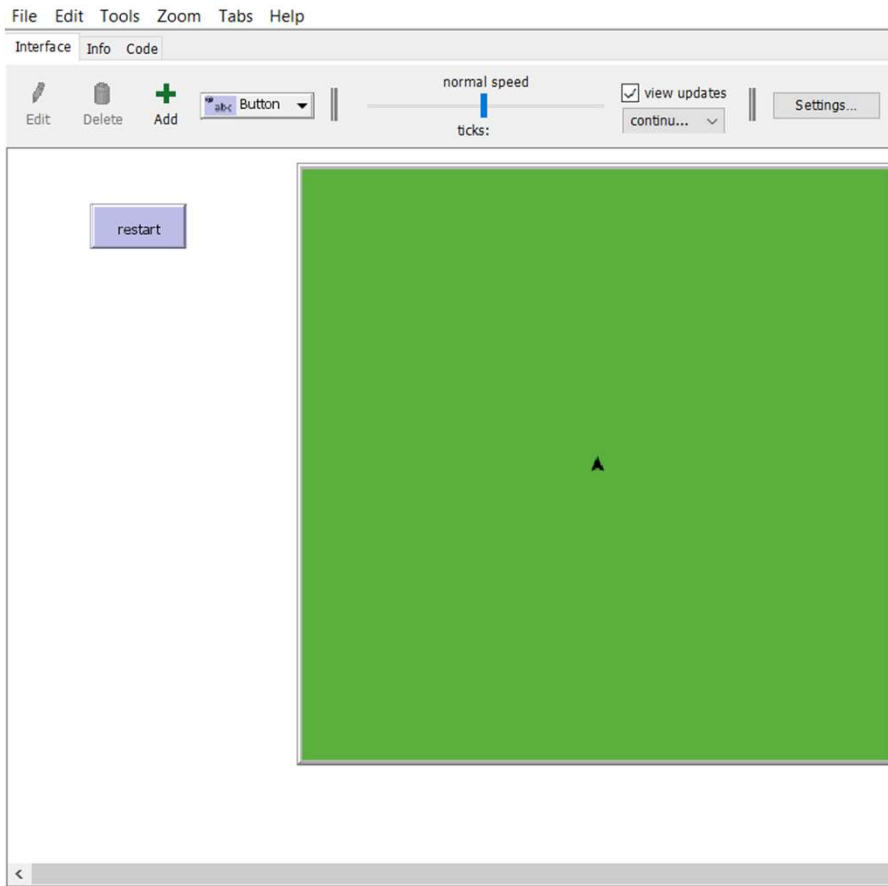
```
observer> create-turtles 20
observer> ask turtles [forward 10]
```

**If we enter:  
*ask n-of 10 turtles [die]***

**What will happen?**

***NOTE: NetLogo automatically randomizes!***

# Basic patch commands



**Switch to patches>**

```
Command Center  
patches> set pcolor green
```

**Note: as observer, we can ask patches similar things as turtles, except patches do not move**

# Let's create a simulation model: Tab Code

```
breed [sheep a-sheep]
```

```
to setup
```

```
  clear-all
```

```
  ask patches [ set pcolor green ]
```

```
  set-default-shape sheep "sheep"
```

```
  create-sheep initial-number-sheep [
```

```
    set color white
```

```
    setxy random-xcor random-ycor
```

```
  ]
```

```
  reset-ticks
```

```
end
```

```
to go
```

```
  if not any? turtles [ stop ]
```

```
  ask sheep [
```

```
    move
```

```
    death
```

```
  ]
```

```
end
```

```
to move
```

```
  rt random 50
```

```
  lt random 50
```

```
  fd 1
```

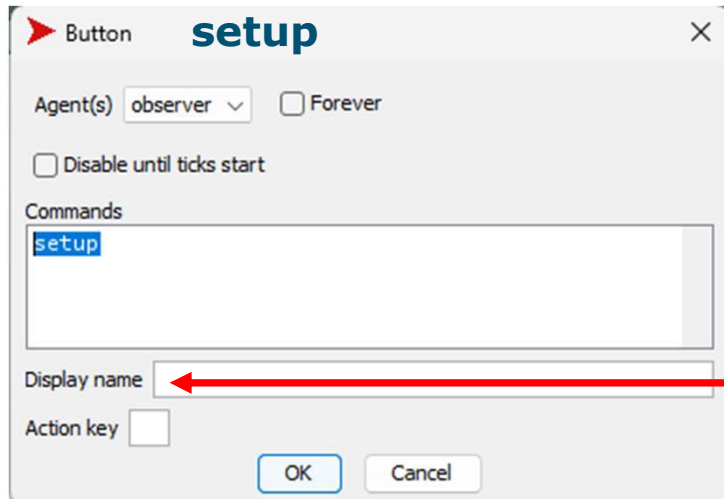
```
end
```

```
to death
```

```
  if random-float 1 < 0.01 [ die ]
```

```
end
```

# Interface: we need to make buttons and sliders



Button **setup**

Agent(s)  ☐ Forever

☐ Disable until ticks start

Commands

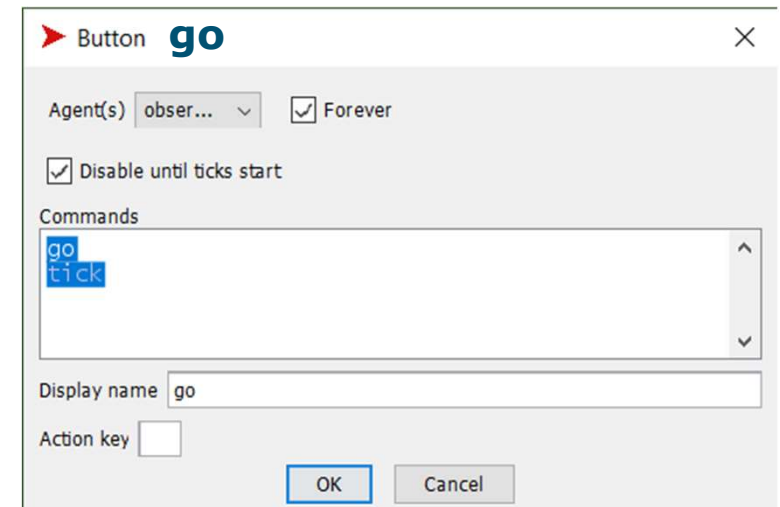
`setup`

Display name

Action key

OK Cancel

Give it a name



Button **go**

Agent(s)  ☒ Forever

☒ Disable until ticks start

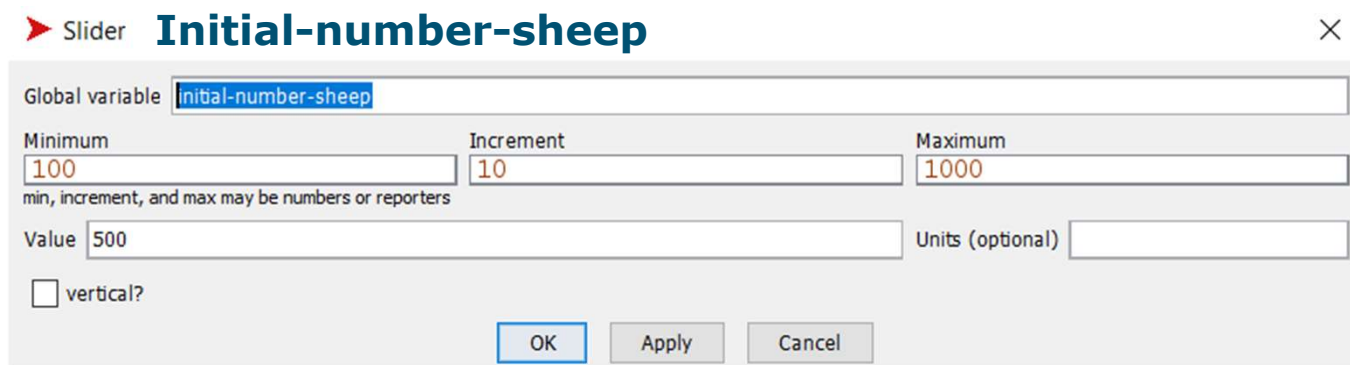
Commands

`go`  
`tick`

Display name

Action key

OK Cancel



Slider **Initial-number-sheep**

Global variable

Minimum  Increment  Maximum

min, increment, and max may be numbers or reporters

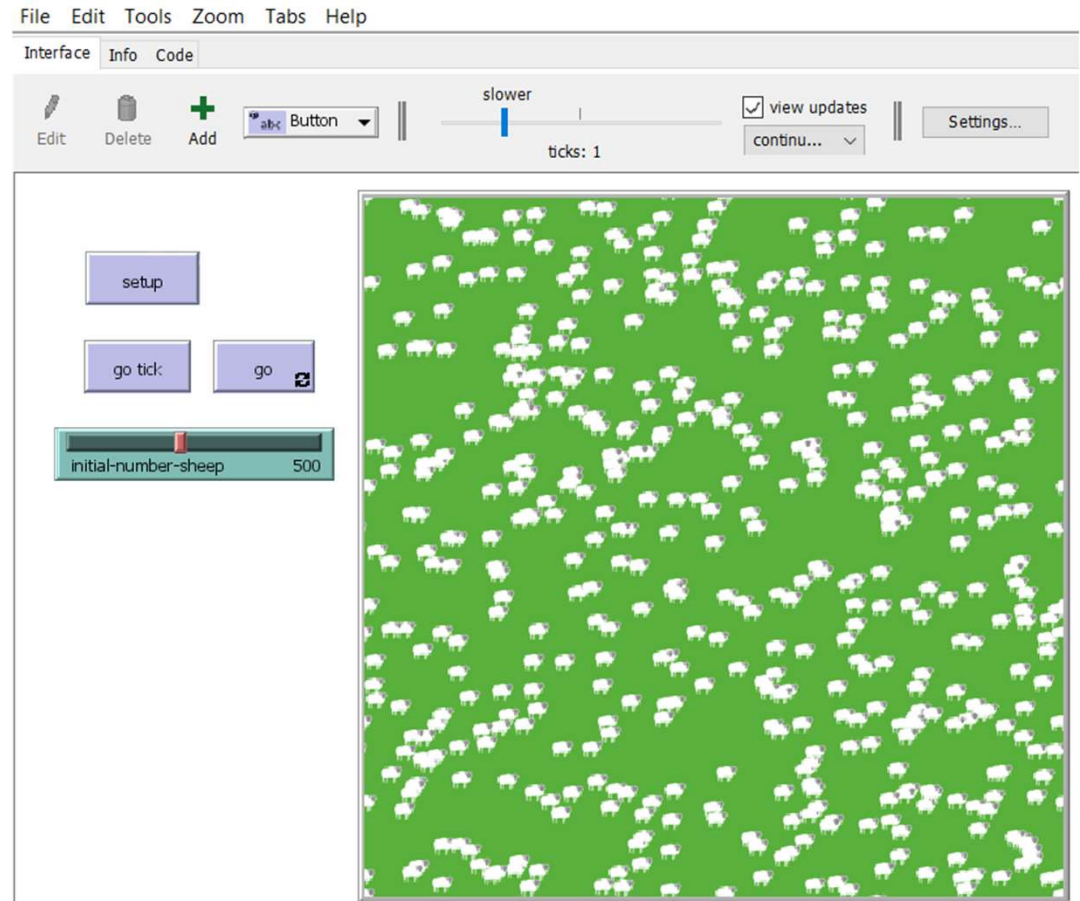
Value  Units (optional)

☐ vertical?

OK Apply Cancel

# Model results and comments

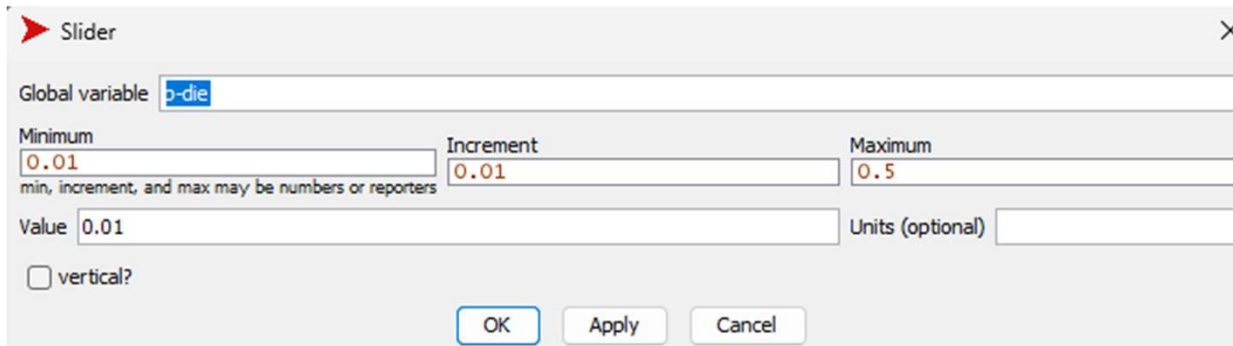
- Press 'Go'
- All sheep eventually die
- Good practice would be to:
- Not hard-code parameters like the probability of dying but make it a slider
- To provide a flow diagram



# Making parameters as sliders

to **death**

```
if random-float 1 < p-die [ die ]  
end
```



Slider

Global variable

Minimum  Increment  Maximum

min, increment, and max may be numbers or reporters

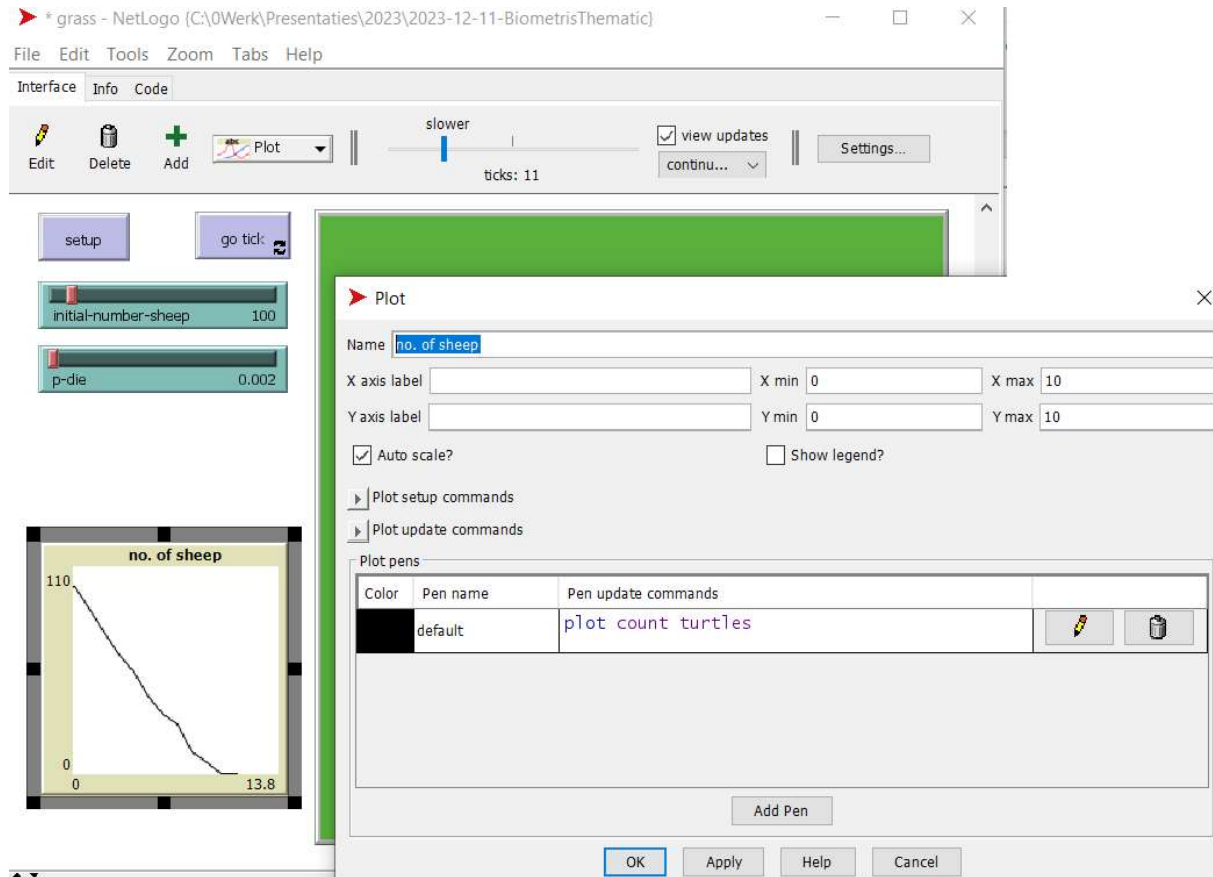
Value  Units (optional)

☐ vertical?

OK Apply Cancel

- Think of which program variables in your code are parameters that can be turned into sliders (or switches)
- ➔ sensitivity analysis with BehaviorSpace

# Make a plot



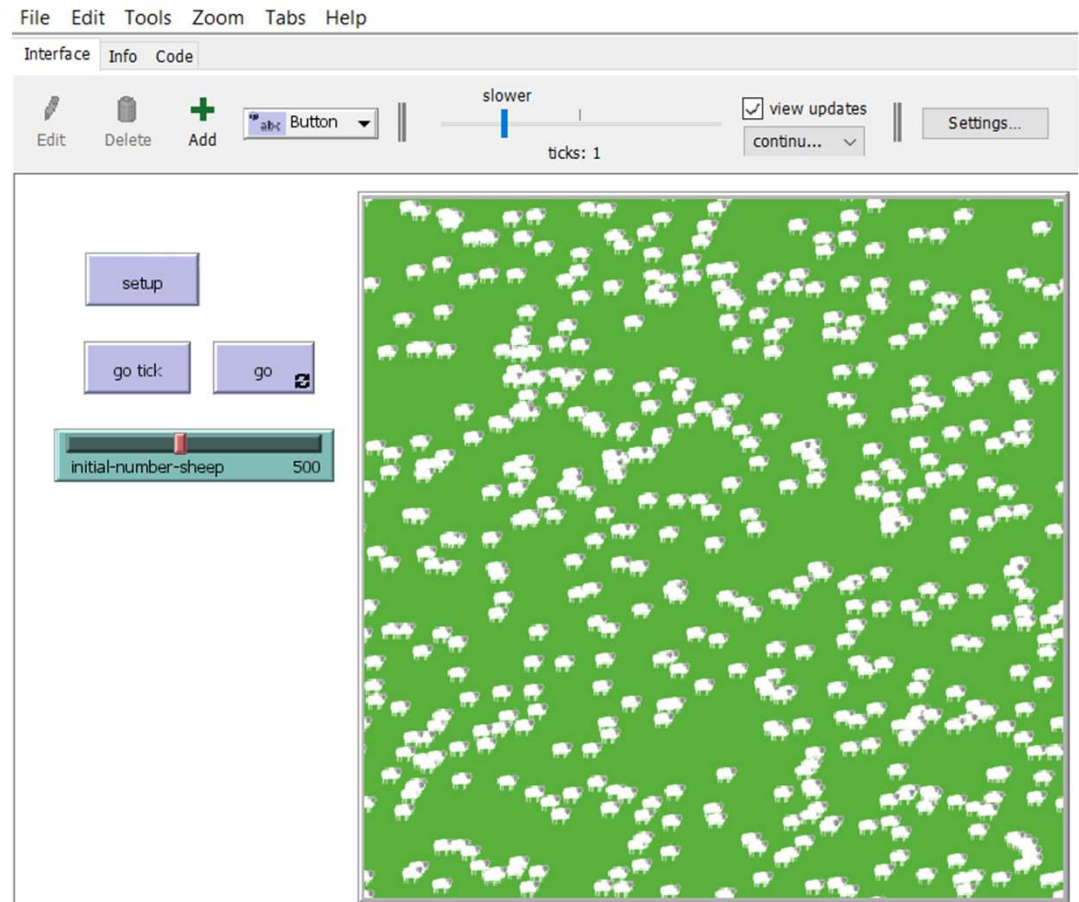
**Save your sheep model**

**We will edit it multiple times**



# Expanding the action space

- All sheep die
- Now how to breed new sheep?
- Hint: the command is **“hatch”**



# Assignment: Have new turtles being born

- E.g., probability-driven

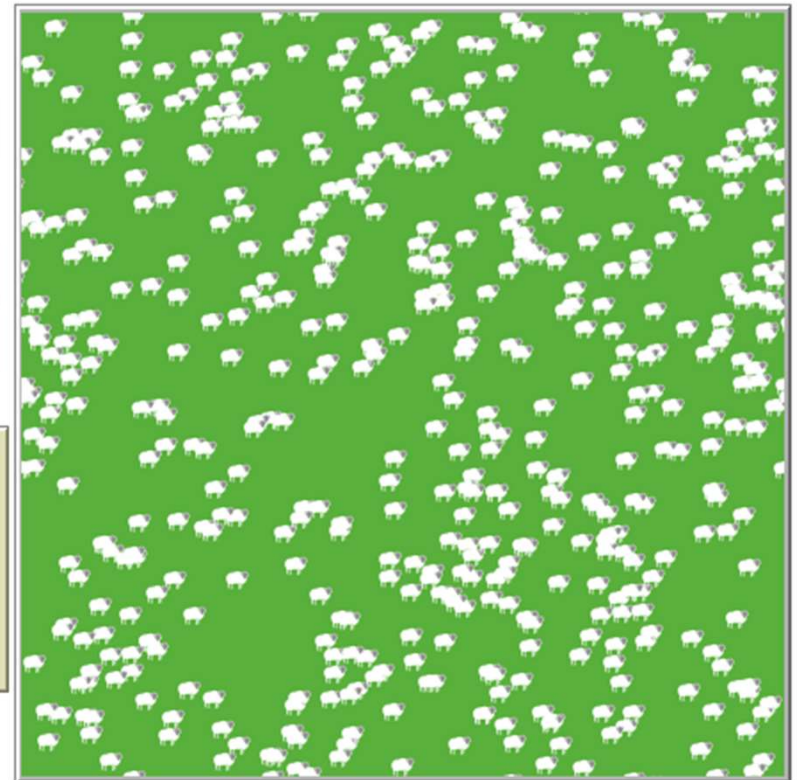
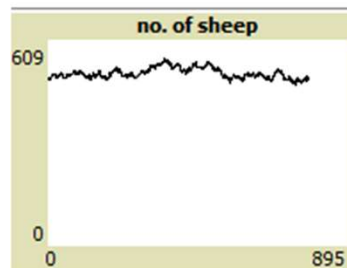
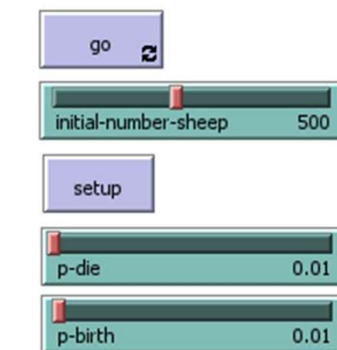
```
to go
  if not any? turtles [ stop ]
  ask sheep [
    move
    death
    born
  ]
end
```

*The 'decision' to breed is  
now purely stochastic*

```
to born
  if random-float 1 < p-birth [ hatch 1 ]
end
```

# Steady state: input = output

- If  $p\text{-die} = p\text{-birth}$ , we mimic a stochastic steady state



# Small assignment: informed decision to breed

- Revamp the sheep model to do the following:
  - 1/ There is an influx of new sheep each time step
  - 2/ Sheep die if the local sheep density is too high
- Hints:
  - Again, use command “hatch” in a submodule ‘born’
  - Command: in-radius  $X$ , with  $X$  a number
  - Use “let” to set a local counter

```
to sheep-observe
  let sheep-count count sheep in-radius X
[...]  
end
```

# A possible model

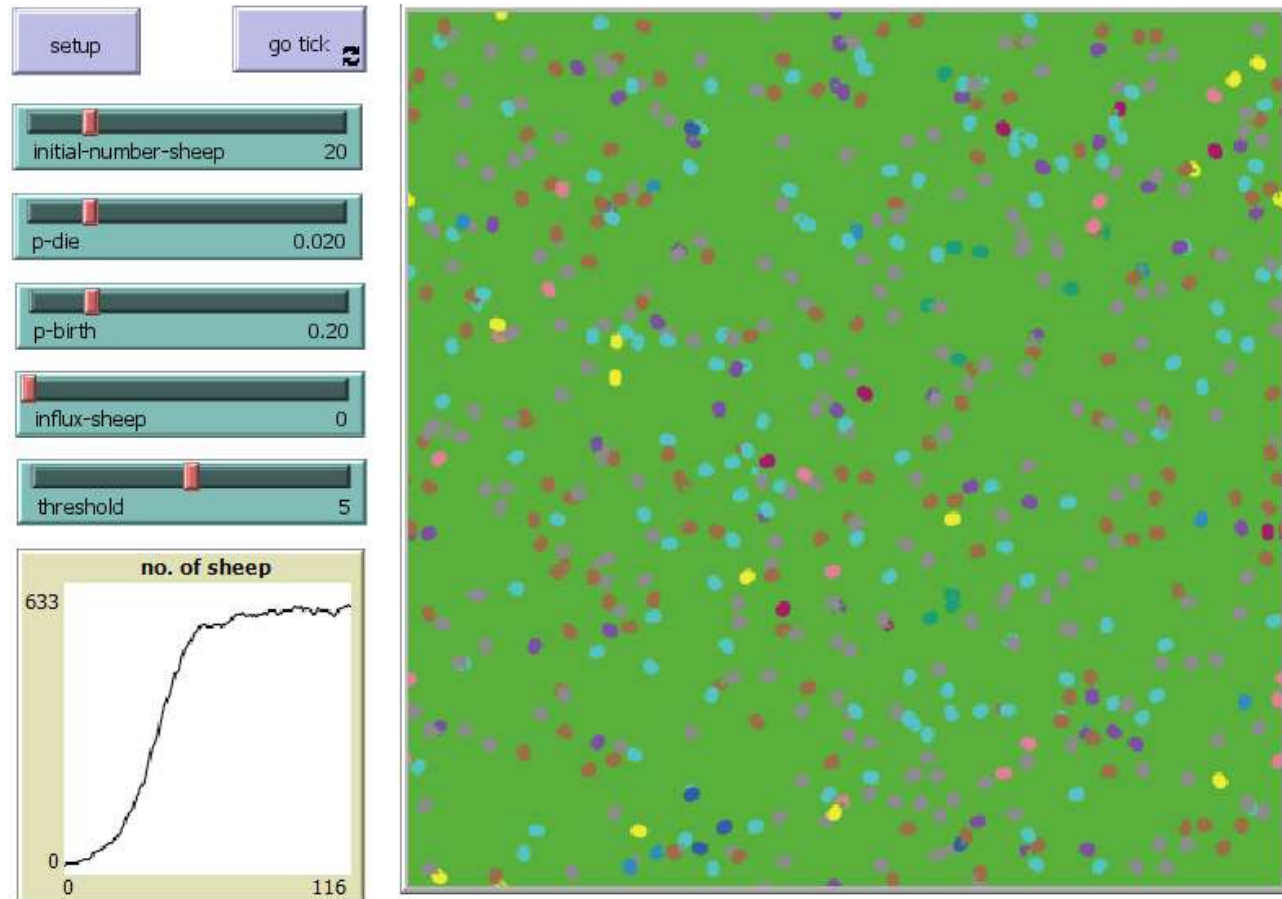
- Influx of new sheep:
  - **create-sheep**
- Observe neighbouring spaces:
  - **sheep-observe**
  - Density-dependent birth and death with **threshold**

*The 'decision' to breed or die is now informed by external factors*

```
to go
  if not any? turtles [ stop ]
  ask sheep [
    sheep-observe
    move
  ]
  create-sheep influx-sheep [
    set color white
    setxy random-xcor random-ycor
  ]
end

to sheep-observe
  let sheep-count count sheep in-radius 2
  if sheep-count >= threshold [ death ]
  if sheep-count < threshold [
    if sheep-count > 1 [ born ]
  ]
end
```

# Output of the possible model



# Some notes

- You hopefully have figured out by now you can:
  - Stop the simulation by pressing the go-button again
  - Move the buttons, etc. across your screen by mouse right button, 'select', and then move with your mouse
  - You can change the sliders during the simulation
  - You cannot run the simulation until basic errors have been fixed (look for typo's, division by zero, incorrect pointers, etc.)
  - Comments are denoted by `;;`
- We now have a basic sandbox to play

# Intermezzo

- After the break:
  - Resource dynamics
  - Memory
  - Direct interaction links
  - Assignment: multi-agent optimal (?) resource collection model



# Resource dynamics (concept)

- We start a new model (***but save your sheep model for later!***):
- Diffusion of a common pool resource  $u[x, y, i]$
- Diffusion rate  $D$
- Discrete time-stepping with  $\Delta t$  ; tick set to 1
- Discrete space-stepping with  $\Delta x, \Delta y$  ; size of patch set to  $1 \times 1$
- As a patch, you lose material to four neighbours:  **$-4D \cdot u[x, y, i]$**
- You also gain material from those same neighbours:  
 $D(u[x + 1, y, i] + u[x, y + 1, i] + u[x - 1, y, i] + u[x, y - 1, i])$

# Resource dynamics (in NetLogo, 1)

```
globals [  
  deltat  
  ;;  $\Delta x = \Delta y$   
  deltax  
]
```

```
patches-own [  
  u  
  u-old  
]
```

```
to go  
  p-diffuse  
  p-recolor  
  tick  
end
```

```
to setup  
  clear-all  
  ask patches [ set u (random-float K) ]  
  p-recolor  
  set deltat 1  
  set deltax 1  
  reset-ticks  
end
```

Not done yet, we need the actual diffusion...

# Resource dynamics (in NetLogo, 2)

## to **p-diffuse**

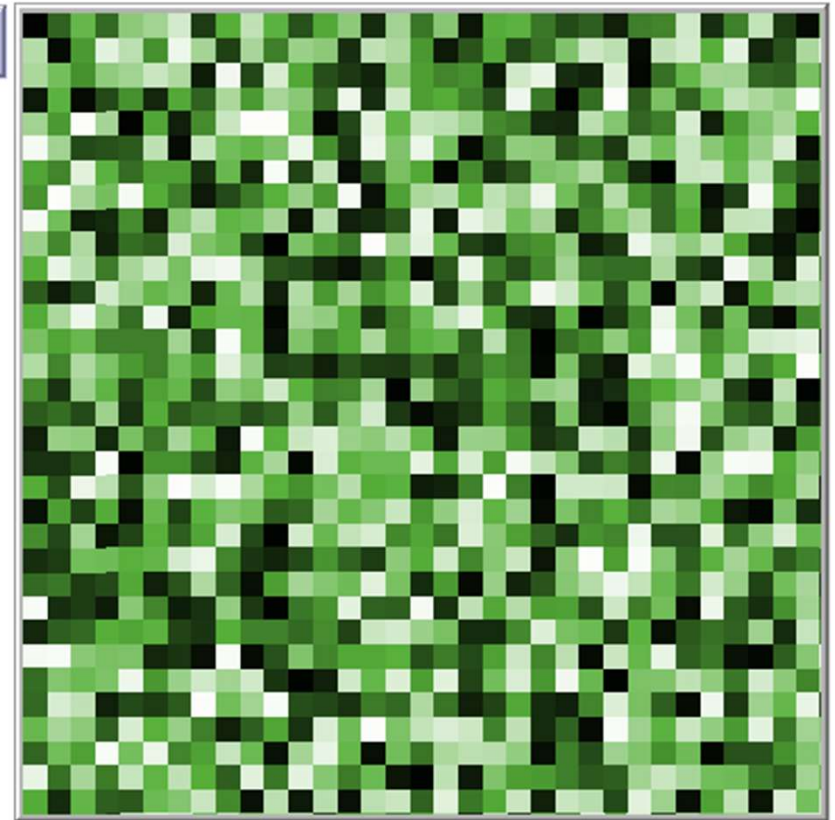
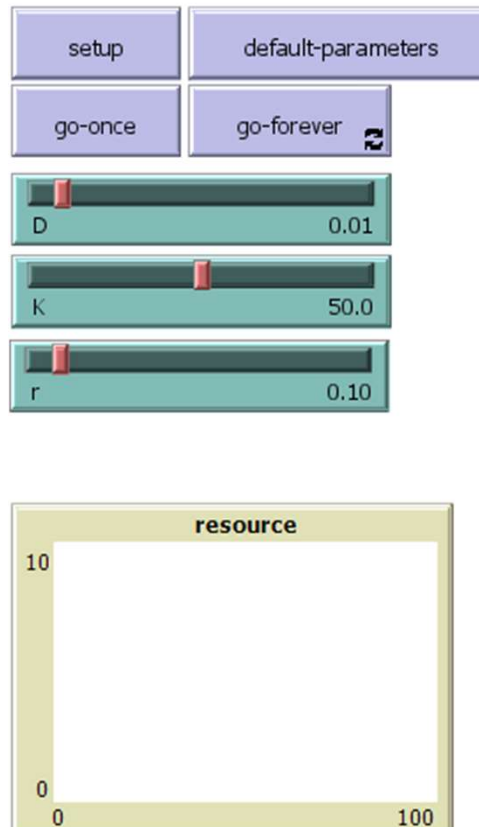
```
;; Store old values, because you need them  
ask patches [ set u-old u ]  
;; Then new values are:  
ask patches [ set u  
  ;; What remains after loss to your four neighbours  
  (1 - count neighbors4 * D) * u-old  
  ;; What you gain from those neighbours  
  + D * sum [ u-old ] of neighbors4  
]  
end
```

## to **p-recolor**

```
ask patches [ set pcolor scale-color green u K 0 ]  
end
```

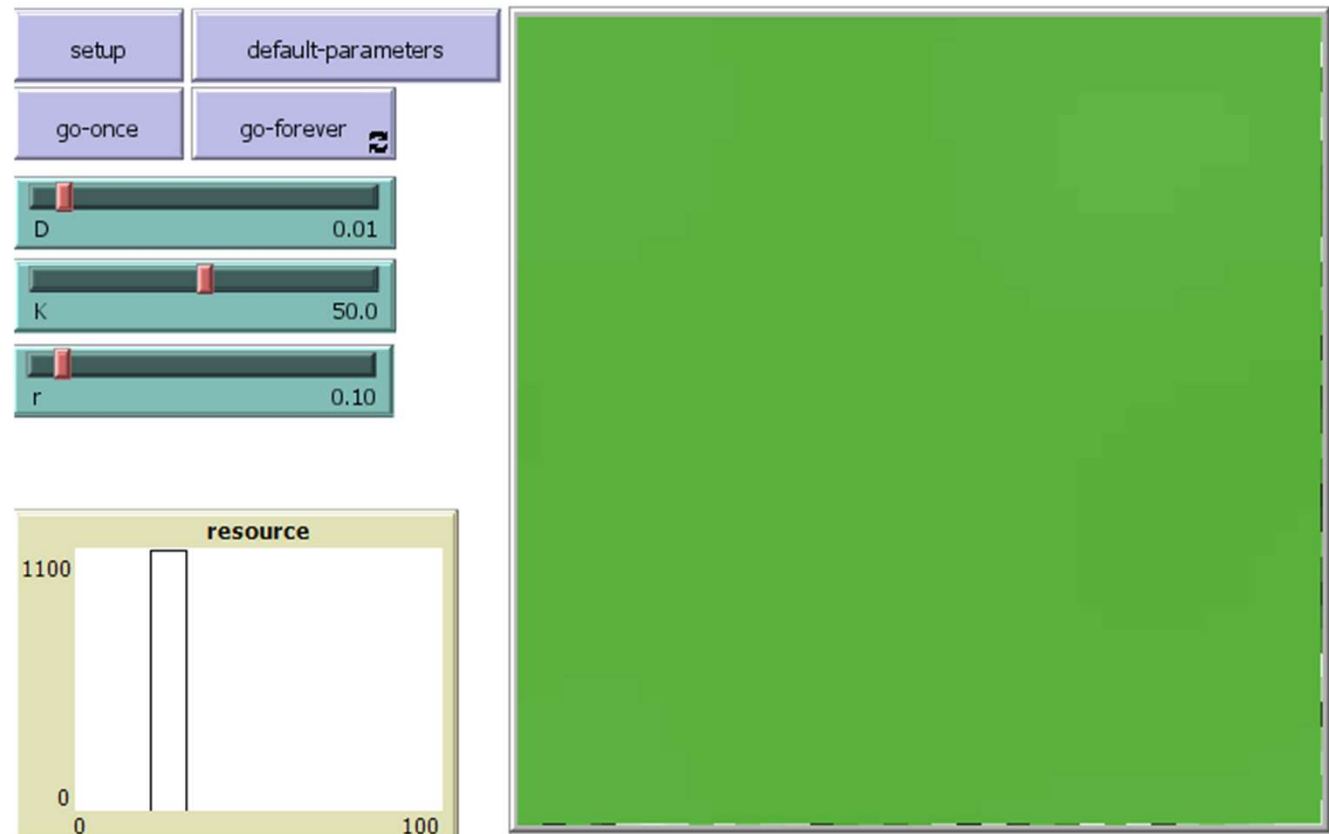
# Running resource dynamics

- Create buttons
- Set slider values
- Run and see what happens



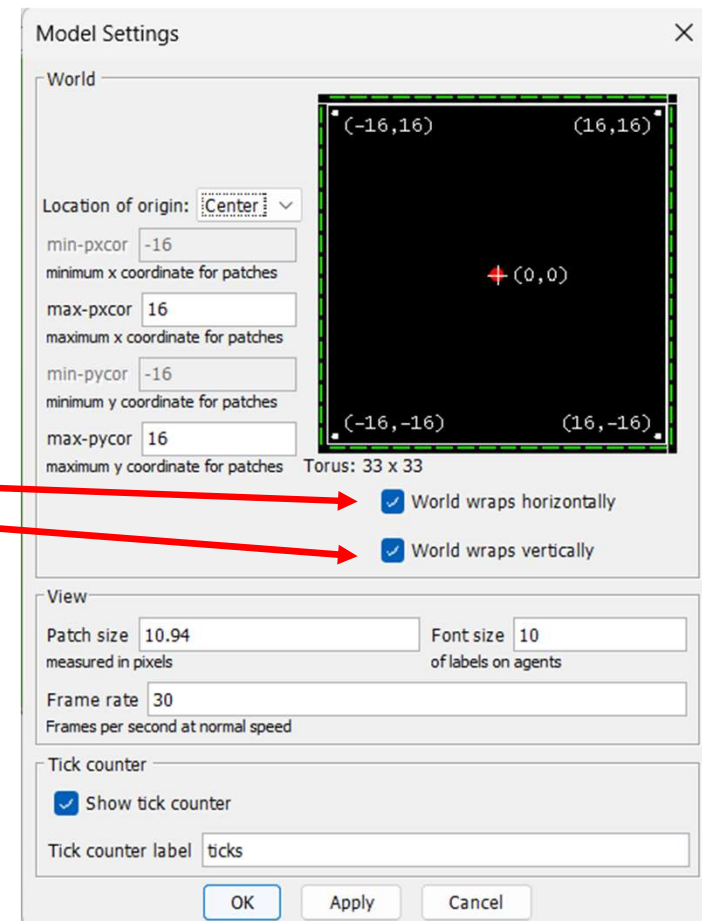
# Resource settles in steady state

- Homogeneous field
- Larger than zero
- Q: Why is not everything gone?



# Changing the world, changing diffusion

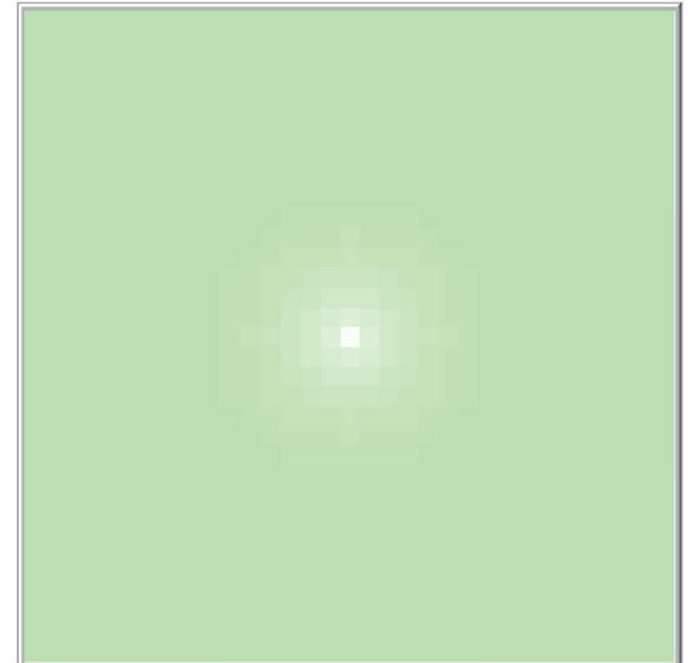
- By default, the world is a torus
- You can change the boundaries using 'Settings'
- Unticking these
- Everything now "runs into the wall"
- So, how do we get leaky boundaries?



# Sinkhole in the middle

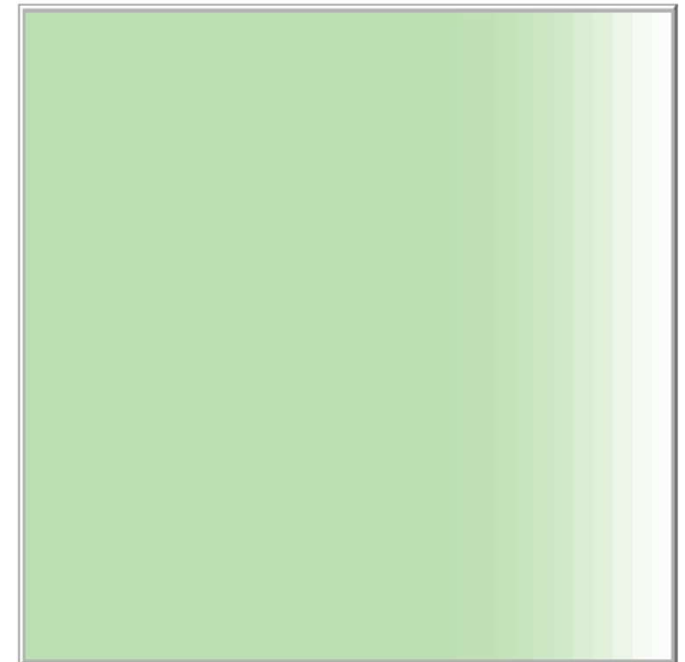
```
to p-diffuse
  ask patches [ set resource-old resource ]
  ask patches [ set resource
    (1 - count neighbors4 * D) * resource-old
    + D * sum [ resource-old ] of neighbors4
    ;; there is a leak in the middle of the field
    ask patch 0 0 [ set u 0 ]
  ]
end
```

The x and y coordinate, respectively



# Leaky boundary at the right

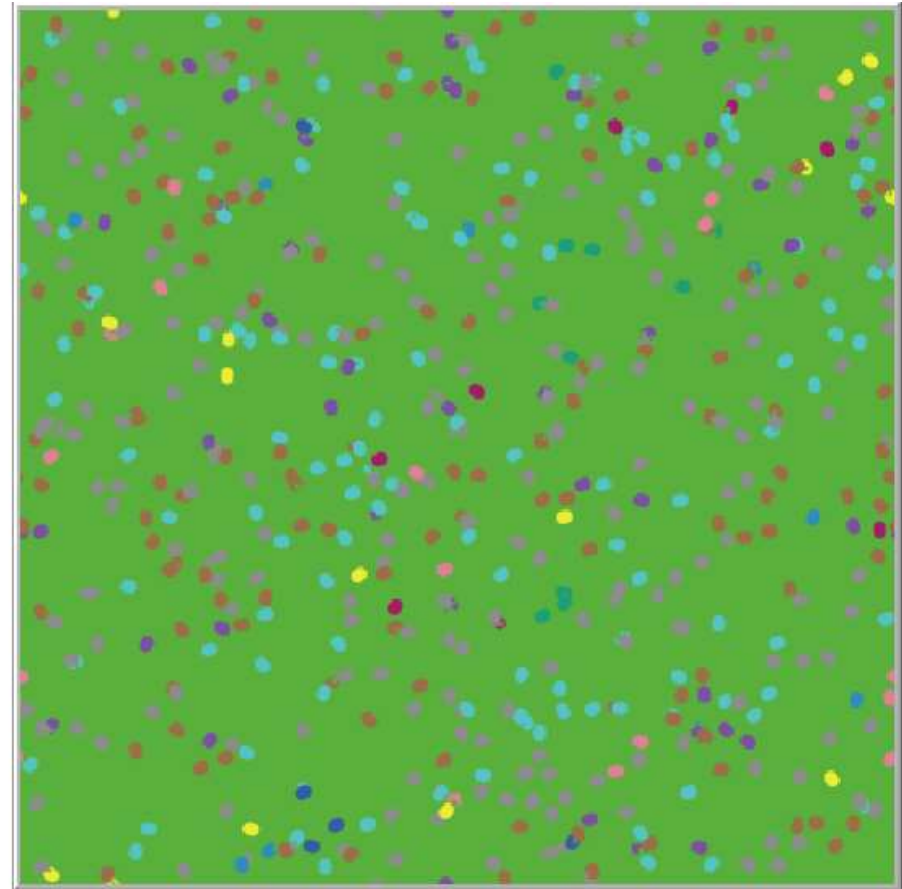
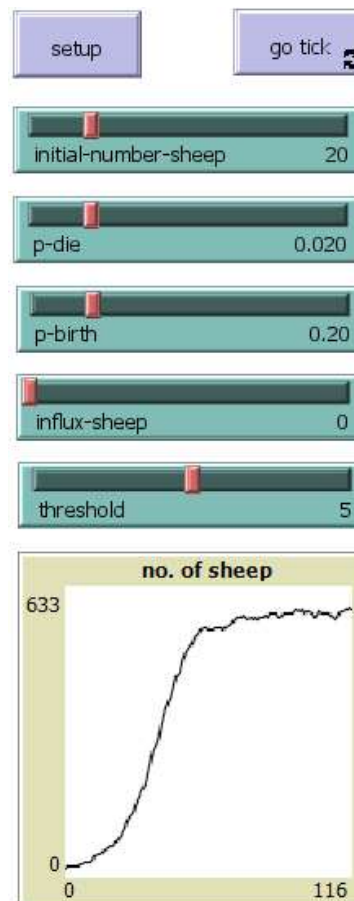
```
to p-diffuse
  ask patches [ set resource-old resource ]
  ask patches [ set resource
    (1 - count neighbors4 * D) * resource-old
    + D * sum [ resource-old ] of neighbors4
    ;; there is a leak at the right side of the field
  ask patches with [ pxcor = max-pxcor ]
    [ set u 0 ]
  ]
end
```





# Back to the sheep model

- Return to your sheep model
- Small assignment:  
**Have your sheep drop off a cliff**



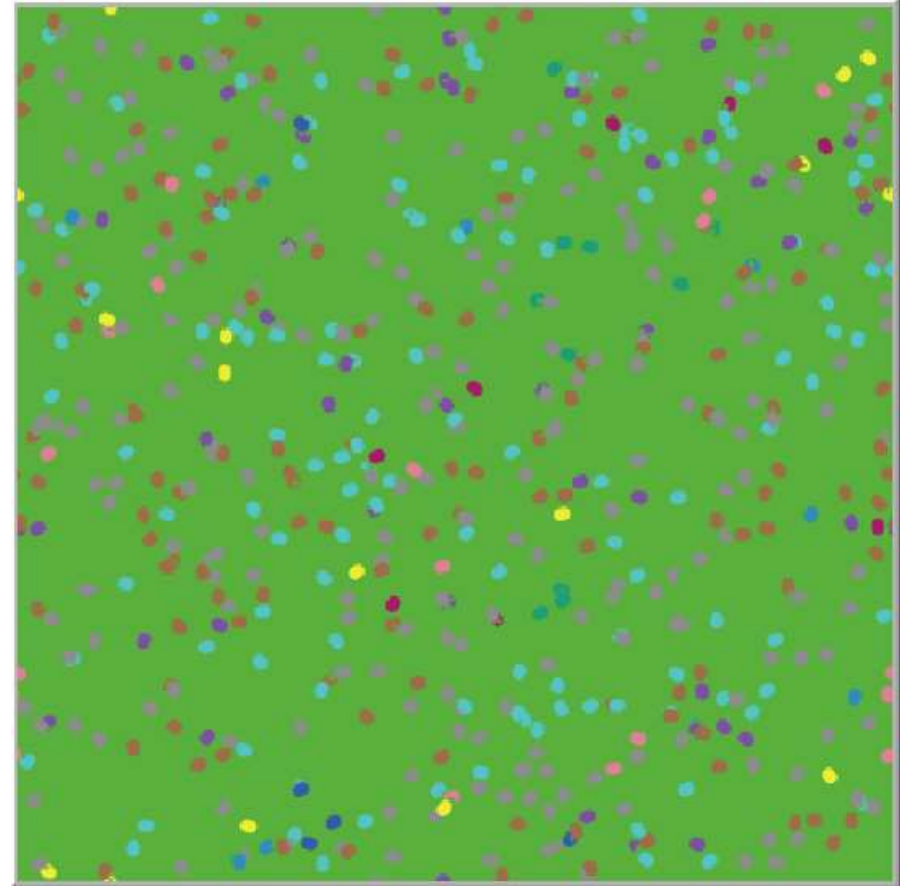
# Sheep diving off a cliff: one solution

```
to move  
  rt random 50  
  lt random 50  
  fd 1  
  ask turtles with [ pxcor = max-pxcor ] [ die ]  
end
```

- You should see them disappear at the far right...

# Agent trait inheritance

- Agents often have characteristics that affect their decisions
- E.g., ecological: internal energy, observed resource and competitors; also, opinion dynamics, stocks, signals,...
- Small assignment:
  - Add some trait inheritance
  - Say, newborn sheep are black



# Newborn sheep are black

```
to born
  if random-float 1 < p-birth [ hatch 1
    [ set color black ]
  ]
end
```

- Eventually all sheep will be replaced by black sheep...
- You can replace this by continuous traits  $w$  with copied (“inherited”) values, e.g.  $w_{new\ agent} = w_{old\ agent} + (random\ error)$

**Under resource scarcity this can recreate natural selection**

# Sheep with money: memory attribute

```
sheep-own [  
  money  
]
```

**to setup** *;; the other commands in this submodule are not shown*

```
  create-sheep initial-number-sheep [  
    set money random-float bigbucks  
    ;; add a slider bigbucks  
  ]  
end
```

**to go** *;; the other commands in this submodule are not shown*

```
  ask sheep [ havemoney ]  
end
```

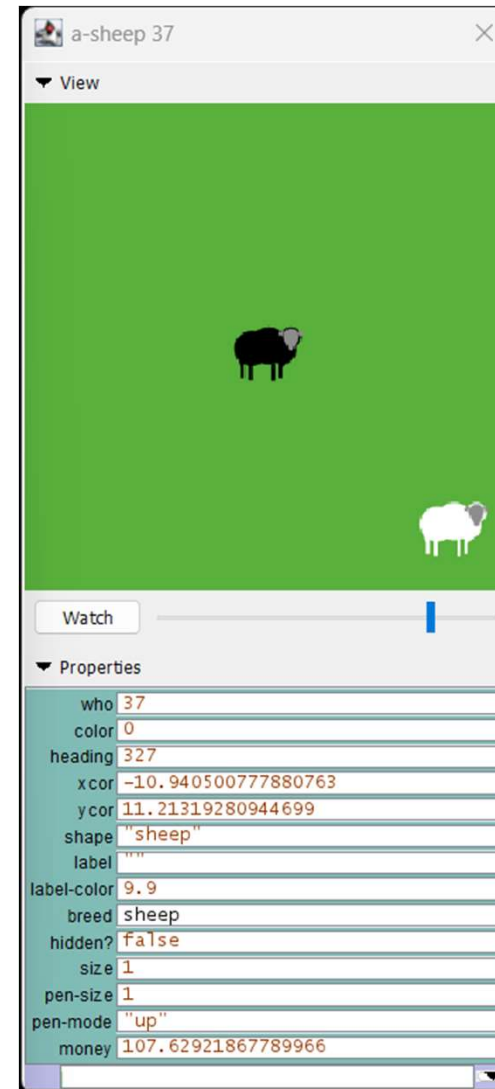
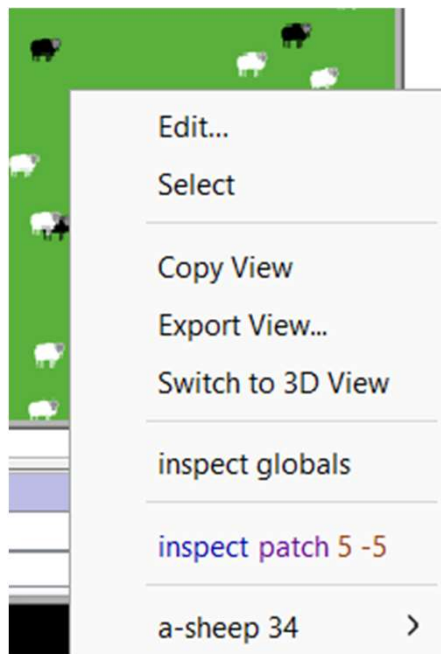
**to havemoney**

```
  set money money + 1  
  if money > 100 [ set color black ]  
end
```

- The sheep will eventually all turn black as they accumulate money

# Inspect your agents

- Right-click on the agent to inspect its properties



# Add a memory: did you pay your taxes? (1)

- Now the full code for the sheep model with paying taxes:

```
breed [ sheep a-sheep ]
```

```
sheep-own [  
  money  
  done?  
]
```

```
to setup  
  clear-all  
  ask patches [ set pcolor green ]  
  set-default-shape sheep "sheep"  
  create-sheep initial-number-sheep [  
    set money random-float bigbucks  
    set done? FALSE  
    set color white  
    setxy random-xcor random-ycor  
  ]  
  reset-ticks  
end
```

## Add a memory: did you pay your taxes? (2)

```
to go
  if not any? turtles [ stop ]
  ask sheep [
    move
    havemoney
    havedone
  ]
end
```

```
to move
  rt random 50
  lt random 50
  fd 1
end
```

```
to havemoney
  set money money + 1
  if money > 100 [ set color black ]
end
```

```
to havedone
  if money > 50 [ paytax ]
end
```

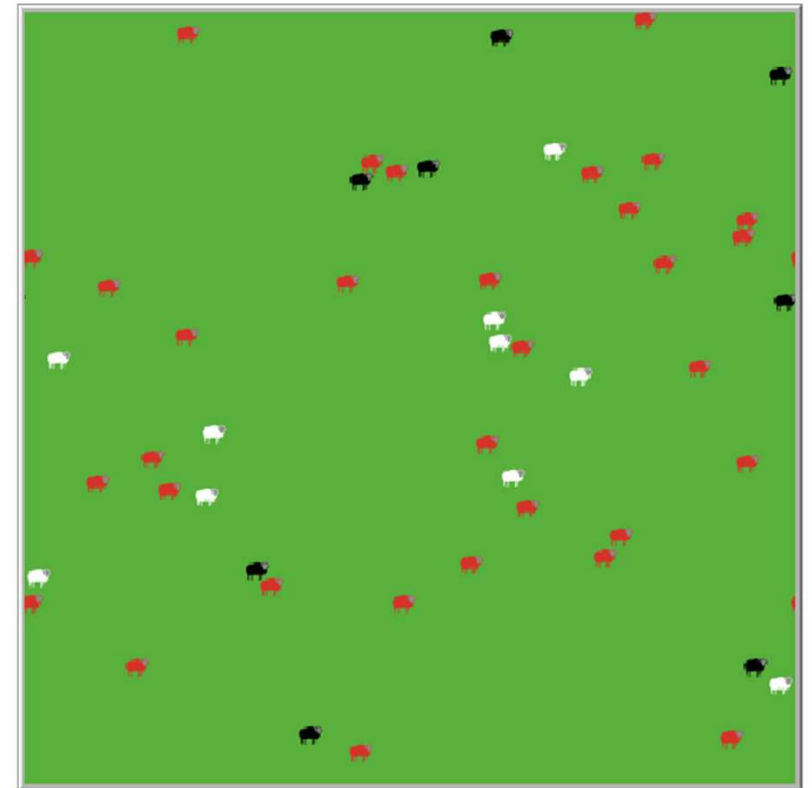
```
to paytax
  if done? = FALSE [
    set money money + 10
    set color red
    set done? TRUE
  ]
end
```

Would be better to make this a slider...



# Check the simulations

- First all sheep are white, then some turn red, later some of the red ones turn black
- Colouring is a good way of checking if steps are executed correctly in your code
- Thus far, agent interactions have been *indirect*



# Creating links (a new type of turtle)

```
to go
  ask sheep [
    move ;; sheep do not breed or die for now
    friend
  ]
end
```

```
to friend
  create-links-to other sheep-here
  ask links [ set color black ]
end
```

- Sheep start linking to other sheep

# Breaking links

## to **setup**

*;; the other commands in this submodule are not shown*

**ask sheep [ create-links-to other sheep ]**

**ask links [ set color black ]**

**end**

## to **friend**

**ask links [ if random-float 1 < p-breaklink [ die ] ]**

*;; so, like agents, links can die*

**end**

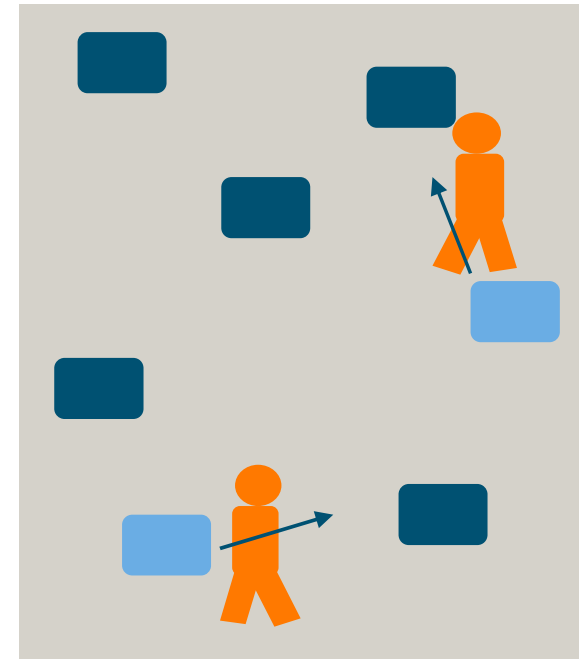
- The number of links gradually disappears
- Note, that the simulation speed increases as well!

# SUMMARY

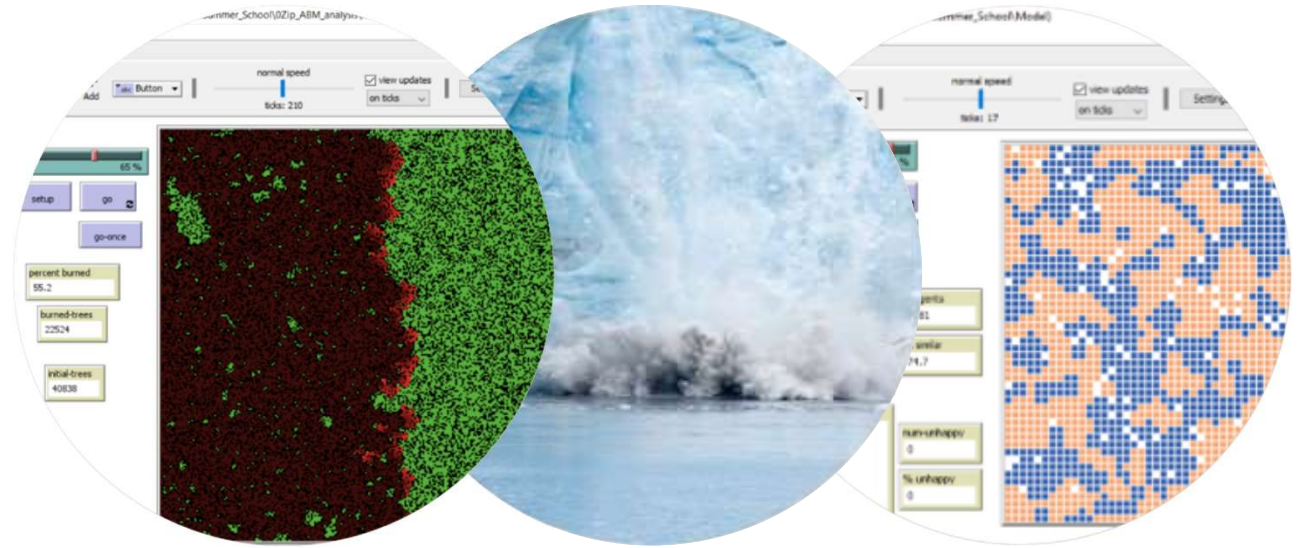
- You have now seen the basics of NetLogo. There are multiple online help sources. Nowadays ChatGTP may be helpful (but...)
- Creative thinking is key
- Cut the code up in small modules that can easily be out-commented
- Think clearly about what you want
  - With the model
  - With the different modules
  - KISSED = Keep It Small, Specific, Engaging, and Doable

# If time permits, an assignment

- Create a simulation model with the following:
- Depletable resource at fixed locations
- Several agents that:
  - Have an internal 'energy meter'
  - Energy slowly depletes
  - Harvesting resource increases energy
  - Risk of death as energy meter decreases
  - Can move at an energy cost
  - First agent to reach a certain threshold "wins"



# Behavioural theories for Stochastic Decision Making



# The rational actor...

- ... Does not exist \*
- Human actors do NOT optimize economic benefit:
  - No perfect information
  - Trade-offs between short-term and long-term gains and loss
  - Choice 'black-out' ( $\geq 3$  options is not possible to evaluate)
  - Subject to non-rational, emotional drivers

# Theory: Human decision making

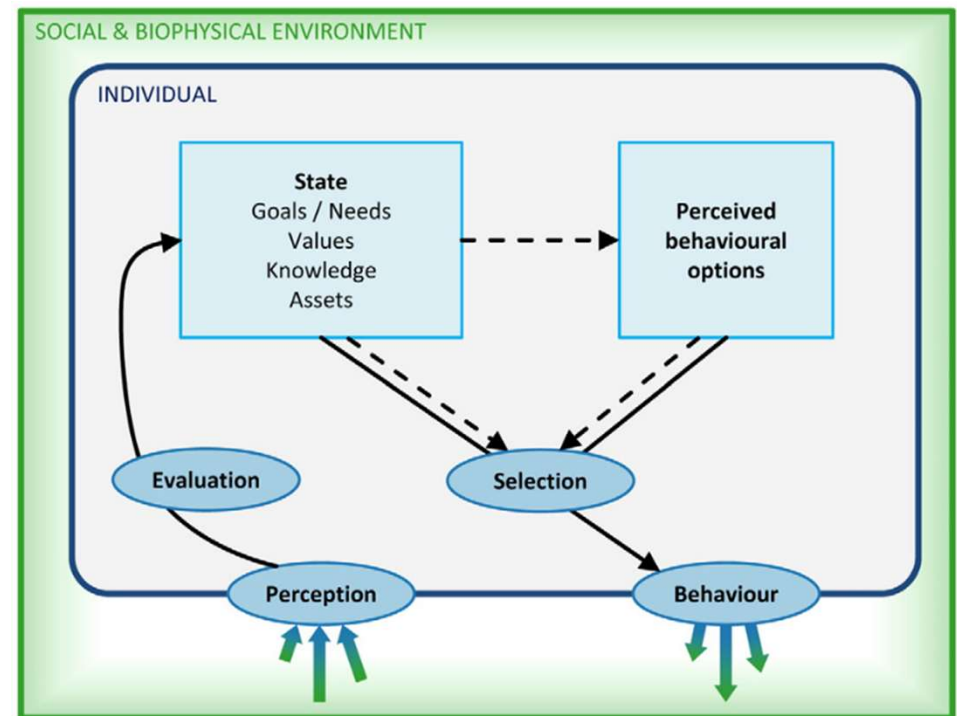
- Some theories you may encounter in social modelling
  - MoHub
  - Grasp / Kemper: status power
  - Six dimensions of culture
  - Consumat
  - Opinion dynamics



# MoHub (Schlüter et al., 2017)

## Q: What drives decisions?

- What goes in?
  - Perception
- What happens in between?
  - Rule checking
- What goes out?
  - Behaviour



# MoHub elements

- Context: social and biophysical environment (e.g., work of Ostrom: shared resources vs. Tragedy of the Commons)
- States: Needs and goals, knowledge, assets, values
- Perceived behavioural options
- Perception (sensing)
- Evaluation
- Selection (could be random!!)
- Actual behaviour

# Status and power (Kemper, Hofstede)

## Q: Why do people do things?

- By **status** ('soft'): voluntary compliance with others
  - Status is a 'resource' you can both claim from another and consume and bestow upon another (e.g., I say 'hi' to you)
- By **power** ('hard'): involuntary compliance with others
  - E.g. through lying, coercion, pleading, or even (non)physical violence
- Reference group: What do others think of you?
  - Motivation = maximizing status

# Cultural dimensions (Hofstede = engineer)

- Nationality / culture drives many human decisions, rooted in:
  - **Individualism**: who / what is important to you?
  - **Power distance**: level of voluntary status conferral
  - **Masculinity**: status conferral based on competitive setting
  - **Uncertainty avoidance**: fear of power of others
  - **Long-term orientation**: status claim from moral issues
  - **Indulgence**: self status conferral by ignoring social rules

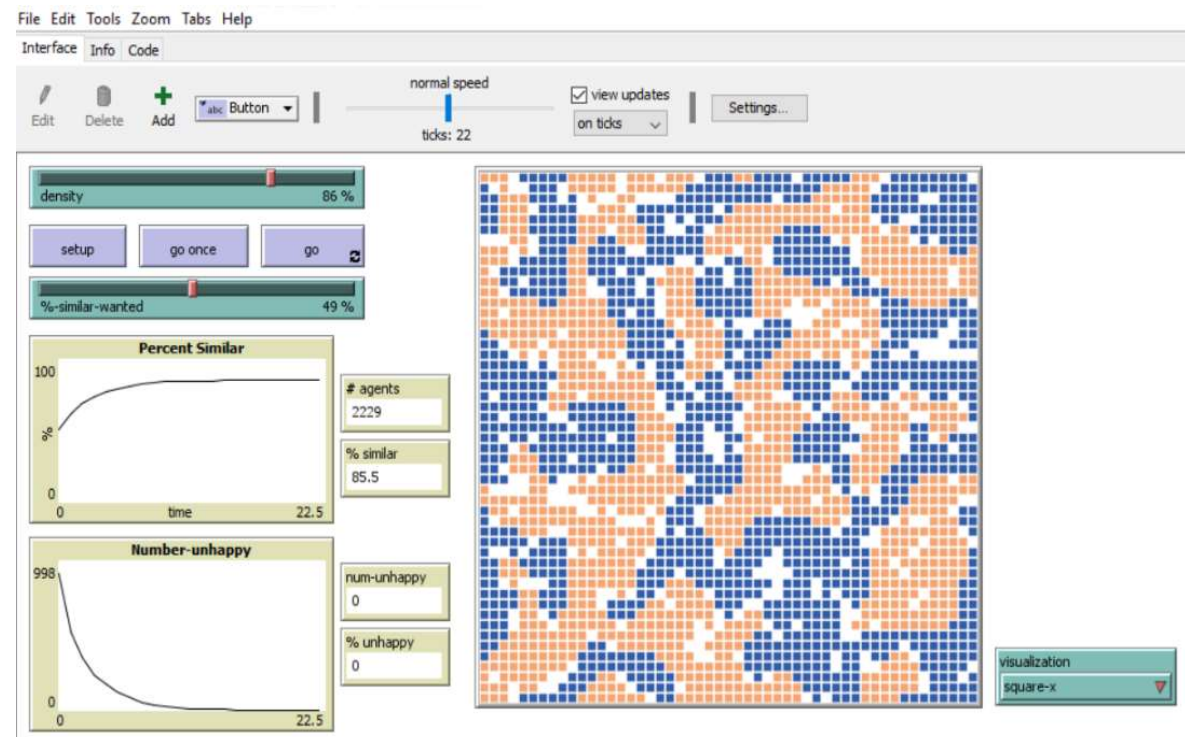
# Consumat (Janssen and Jager)

- No optimizing but 'satisficing'
- Four strategies for selecting behaviour:
  - Low uncertainty and high satisfaction → repetition
  - High uncertainty and high satisfaction → imitation
  - Low uncertainty and low satisfaction → optimizing \*
  - High uncertainty with low satisfaction → inquiry ('social comparison')

# Example: Schelling model (segregation.nlogo)

## Models Library / Social Sciences

- Spaces have one of 3 states (orange, blue, empty)
- Agents desire fraction of neighbourhood to be similar



# Model code

```
globals [  
  percent-similar  
  percent-unhappy  
]
```



The output we monitor

```
turtles-own [  
  happy?  
  similar-nearby  
  other-nearby  
  total-nearby  
]
```



Agent properties:

- Happiness
- Vicinity to other like you

Unhappy turtles want to move elsewhere

# Model code: agent actions

to update-turtles

ask turtles [

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)

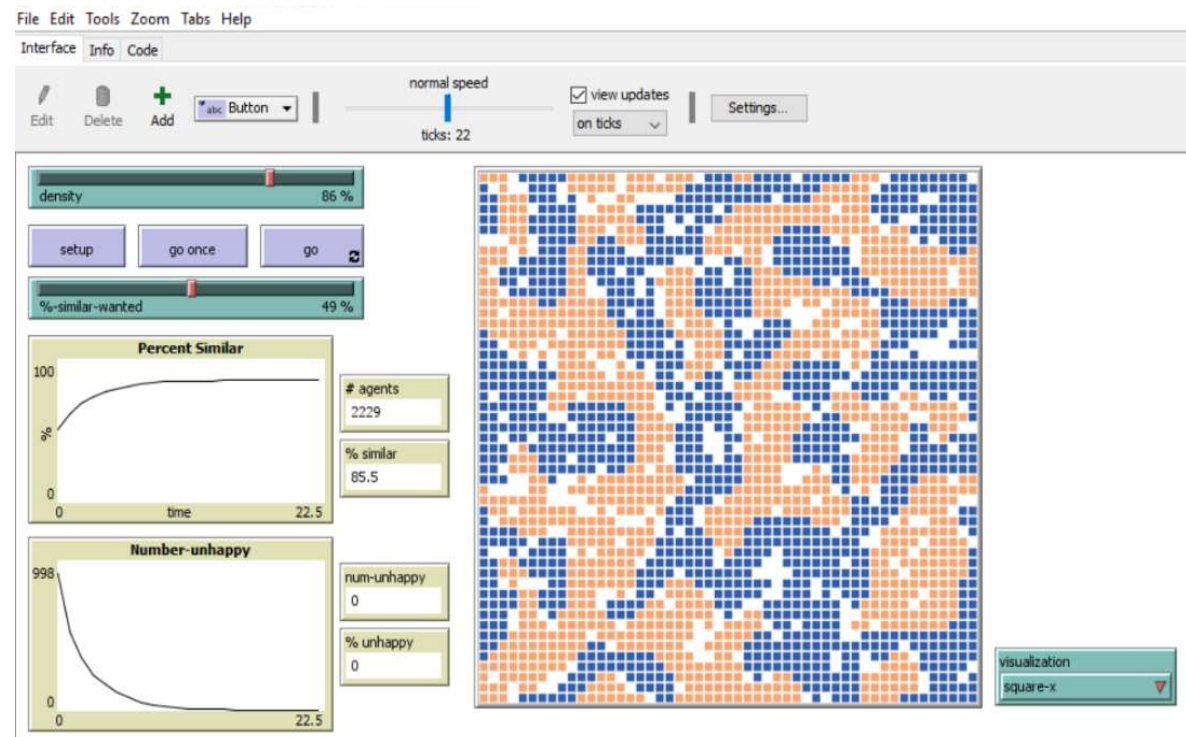
]

end



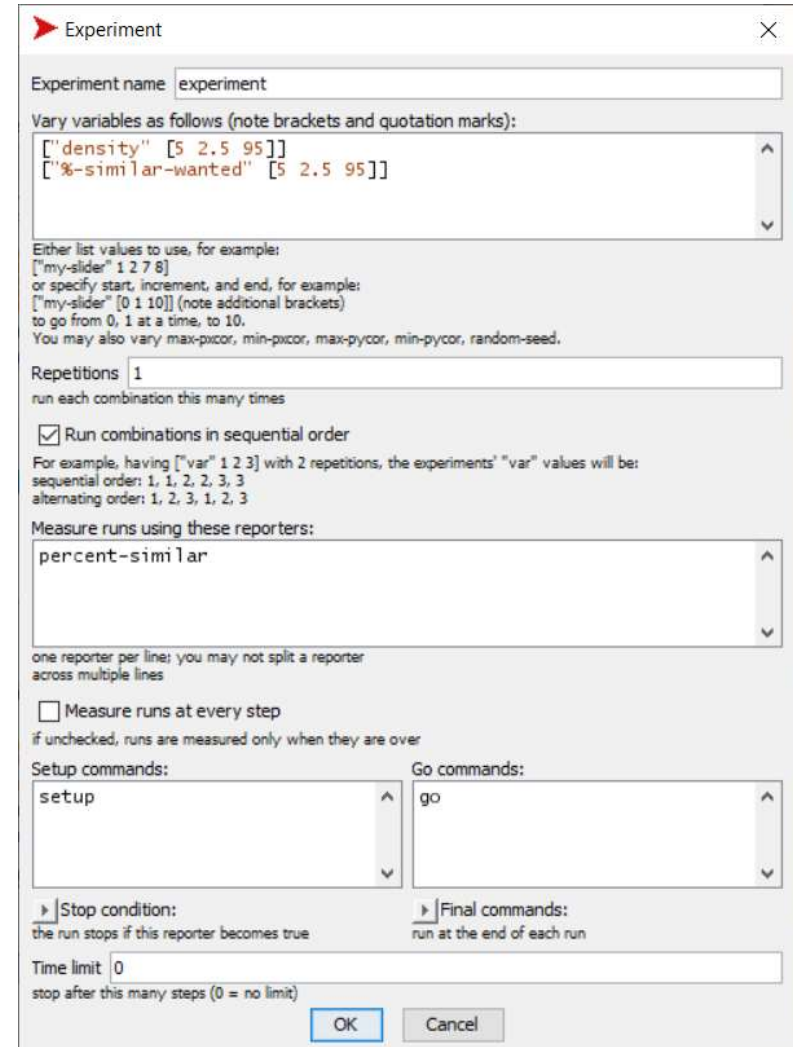
# Model analysis

- Two inputs:
  - density
  - %-similar-wanted
- Two outputs:
  - % similar
  - number-unhappy



# BehaviorSpace

- Two parameters:
- ["density" [5 2.5 95] ]
- ["%-similar-wanted" [5 2.5 95] ]
- Increment of 2.5%
- End: stable spatial pattern
- No replicates
- We have  $(37)^2 = 1369$  runs



The screenshot shows the 'Experiment' configuration window in BehaviorSpace. The 'Experiment name' is 'experiment'. The 'Vary variables as follows' section contains two parameters: ["density" [5 2.5 95]] and ["%-similar-wanted" [5 2.5 95]]. The 'Repetitions' is set to 1. The 'Run combinations in sequential order' checkbox is checked. The 'Measure runs using these reporters' section contains 'percent-similar'. The 'Setup commands' section contains 'setup' and the 'Go commands' section contains 'go'. The 'Stop condition' is set to 'the run stops if this reporter becomes true' and the 'Final commands' are set to 'run at the end of each run'. The 'Time limit' is set to 0.

Experiment

Experiment name: experiment

Vary variables as follows (note brackets and quotation marks):

```
["density" [5 2.5 95]]  
["%-similar-wanted" [5 2.5 95]]
```

Either list values to use, for example:  
["my-slider" 1 2 7 8]  
or specify start, increment, and end, for example:  
["my-slider" [0 1 10]] (note additional brackets)  
to go from 0, 1 at a time, to 10.  
You may also vary max-pxcor, min-pxcor, max-pykor, min-pykor, random-seed.

Repetitions: 1  
run each combination this many times

☒ Run combinations in sequential order

For example, having ["var" 1 2 3] with 2 repetitions, the experiments' "var" values will be:  
sequential order: 1, 1, 2, 2, 3, 3  
alternating order: 1, 2, 3, 1, 2, 3

Measure runs using these reporters:

percent-similar

one reporter per line; you may not split a reporter across multiple lines

☐ Measure runs at every step  
if unchecked, runs are measured only when they are over

Setup commands: setup

Go commands: go

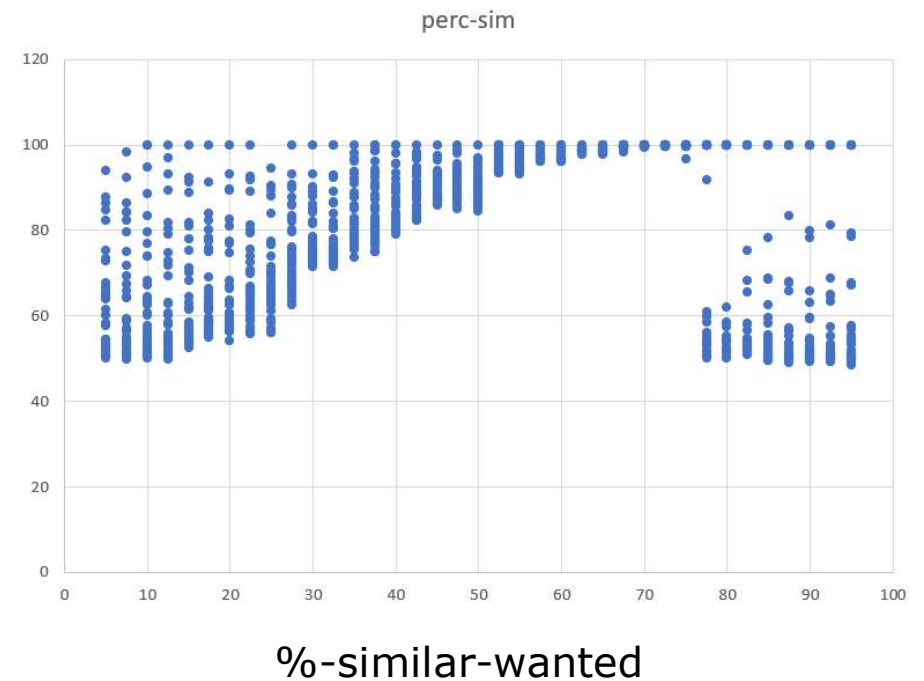
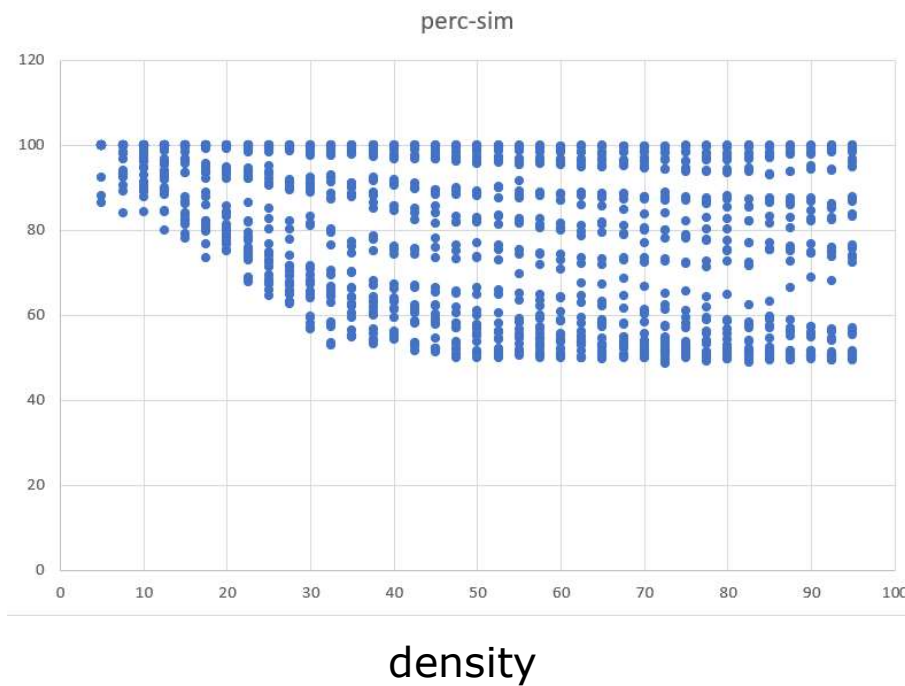
Stop condition: the run stops if this reporter becomes true

Final commands: run at the end of each run

Time limit: 0  
stop after this many steps (0 = no limit)

OK Cancel

# Plots two-parameter output

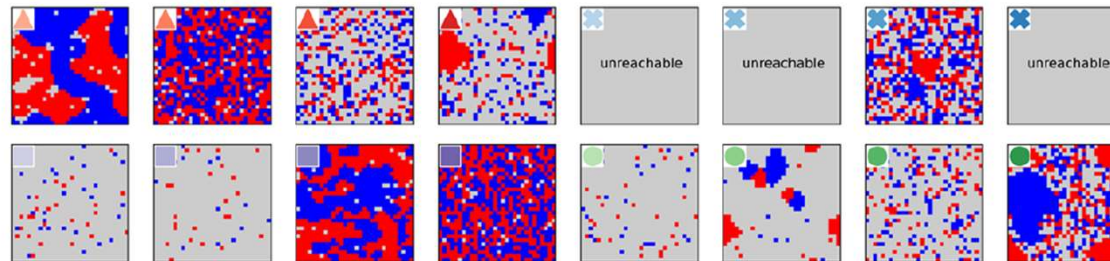
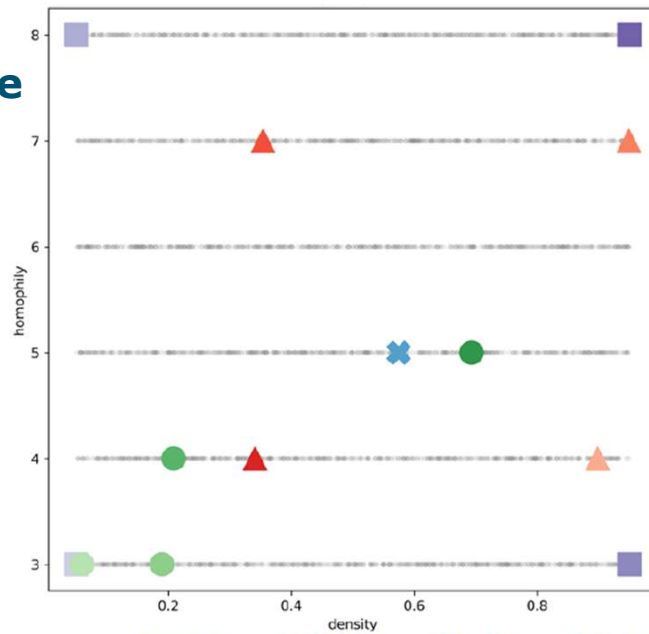


# Segregation model: results

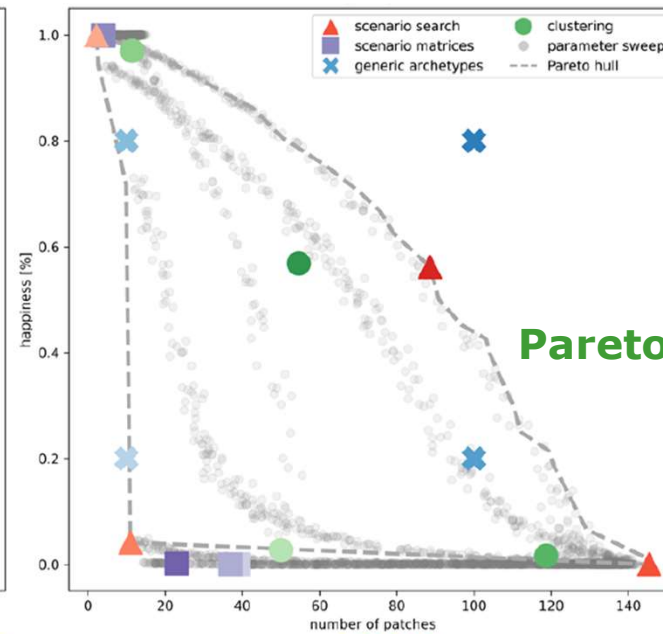
%-similar	Results
<b>Low</b>	Everybody happy, ca. 50% similar, semi-random
<b>Medium</b>	Everybody happy, ca. 75% similar, patterning, fast convergence
<b>High</b>	Everybody happy, > 90% similar, large clustering, slow convergence
<b>Very high</b>	Nobody happy, random, no convergence

# Scenario exploration

Input space



Output space



Pareto front

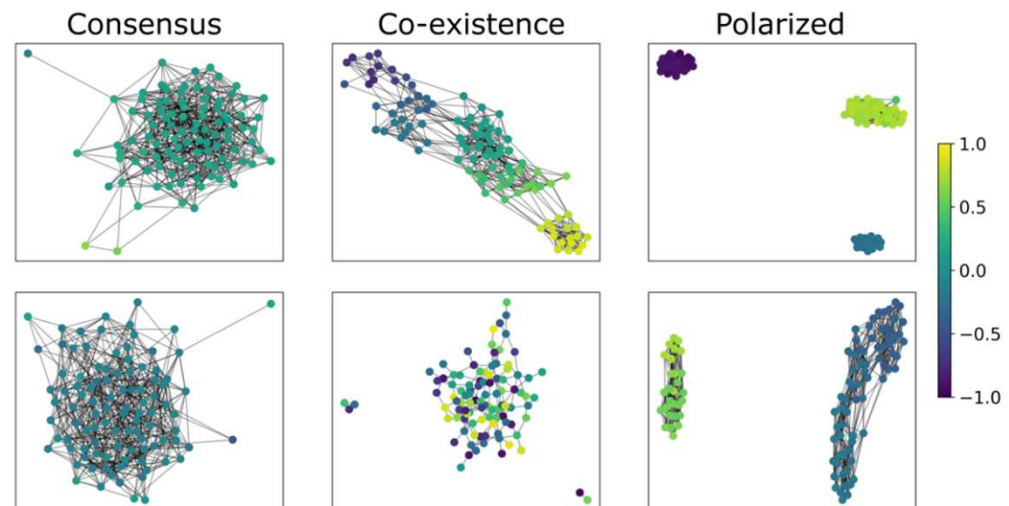
# Schelling model discussion

- Obviously, a model that does not adequately describe reality
- But... Provides a mechanistic hypothesis for observed segregation excluding explicit discrimination by agents
- Multi-objective optimization: model analysis reveals which policy scenarios may be achievable and which ones not

# Opinion dynamics (Weinans et al., 2024)

*Doi: 10.18564/jasss.5212*

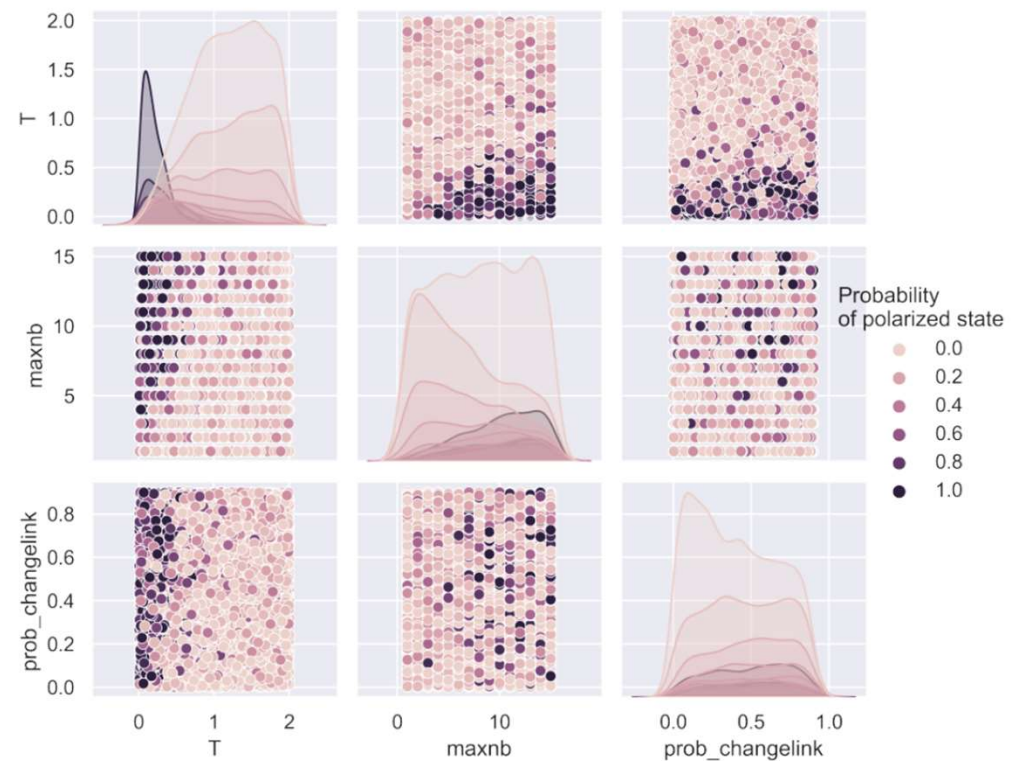
- Adapted Ising model for spin
- Opinions are affected by:
  - Others: fast-changing
  - Our values: slow-changing
  - Our personality: fixed
- Link breaking and relinking
- Different outcomes based on 'settings'





# Opinion dynamics model results

- Results maybe not surprising, but confirming:
- Increasing parameter for distance in opinion for which agents break link → Increased coexistence
- Lower  $T$  = less influence of outside world (e.g., media) → Polarization increases





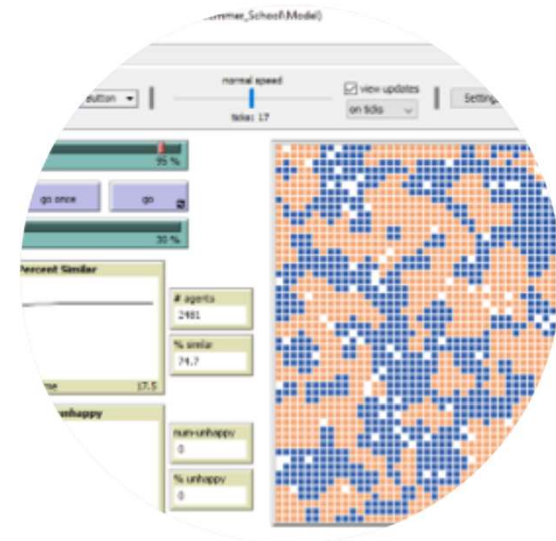
# Social theory in ABM

- All above-mentioned theories have been implemented in ABMs
- Obviously, there are multiple choices to do so
- Calibration and validation is typically problematic
- ABM is a good tool for explorative thinking about stochastic decision making ('what if?'), **not** for predictive quantification

# Thanks!

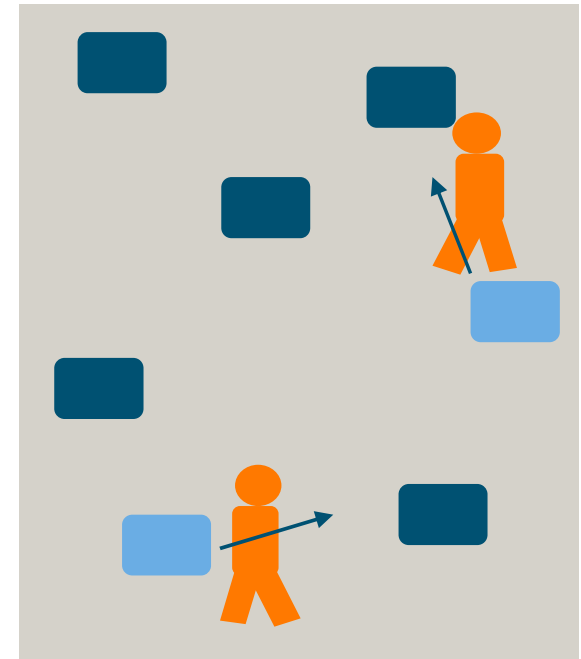
[George.vanVoorn@wur.nl](mailto:George.vanVoorn@wur.nl)

And if time permits, an(other)  
assignment



# If time permits, an(other) assignment (old one)

- Create a simulation model with the following:
- Depletable resource at fixed locations
- Several agents that:
  - Have an internal 'energy meter'
  - Energy slowly depletes
  - Harvesting resource increases energy
  - Risk of death as energy meter decreases
  - Can move at an energy cost
  - First agent to reach a certain threshold "wins"



# If time permits, extend the assignment

- Now extend this with diversity in agent population, following four Consumat strategies for selecting behaviour:
  - Low uncertainty and high satisfaction → repetition
  - High uncertainty and high satisfaction → imitation
  - Low uncertainty and low satisfaction → optimizing
  - High uncertainty with low satisfaction → social comparison
- Q: How do you implement uncertainty and satisfaction?
- Q: Which agent strategy is more successful?