

# Agent-Based Modelling

Computational Social Science  
Workshop 9

# Outline

Work with a simple NetLogo model

- Conformity with majority vote

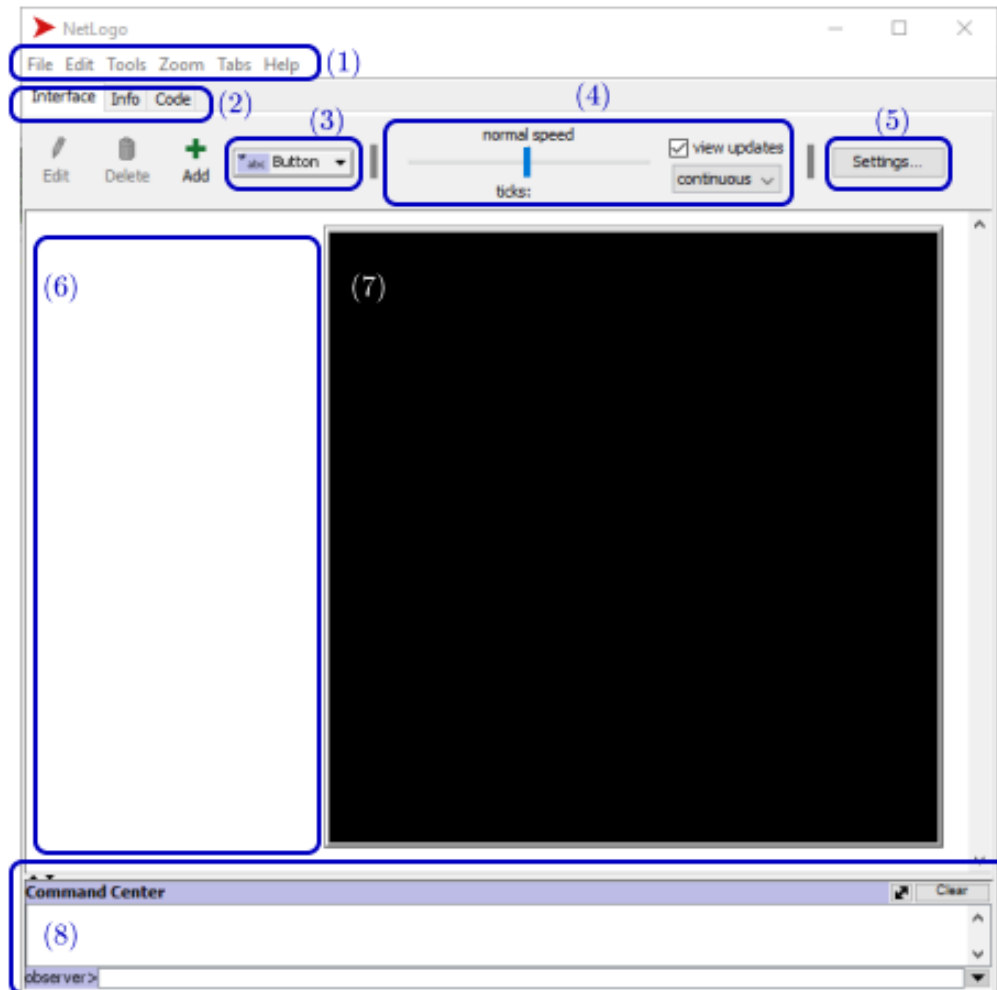
Exercises

- Understand the existing model
- Experiment with the model
- Extend the model

**Understand the  
model**



# NetLogo orientation



1.Menu

2.Tabs for Interface, Info, Code view

3.Adds widgets to interface

4.Speed slider, update controls

5.Settings for world display

6.Screen area for widgets

7.World, main view of model

8.Command Center – for interactive code entry and results

# NetLogo entities

Patches have fixed locations

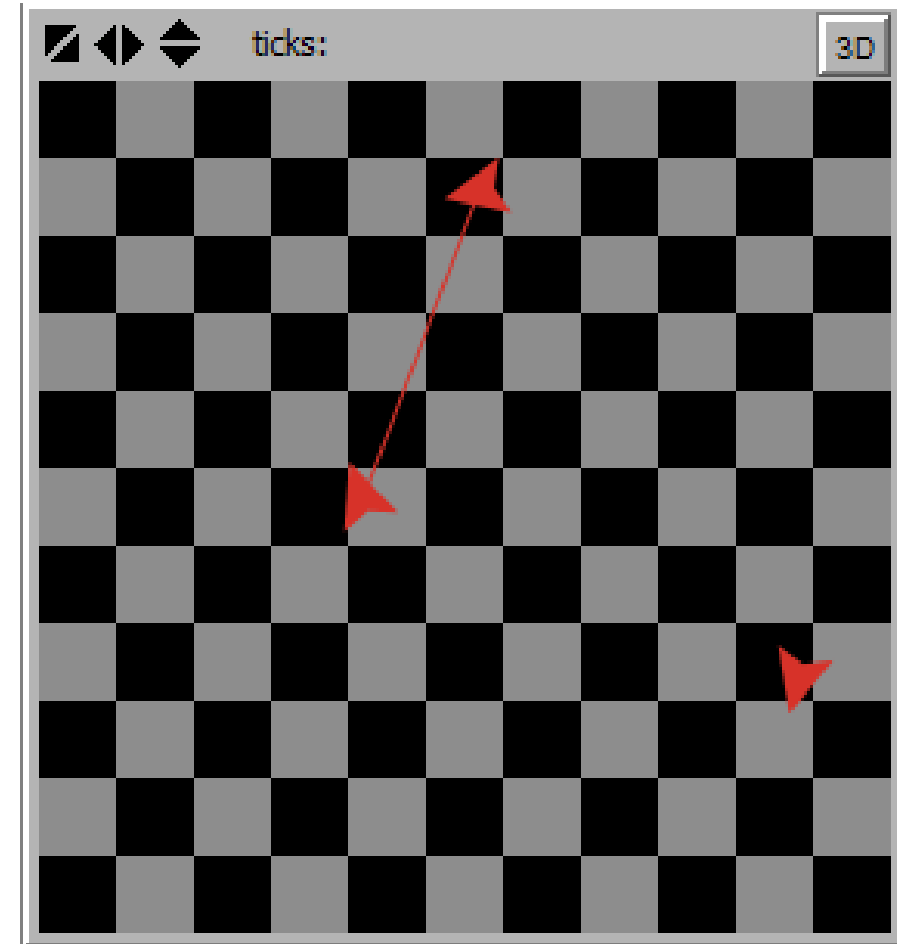
- Black/grey in image
- Typically used for physical environment
- Provide coordinate system for turtle locations

Turtles make decisions

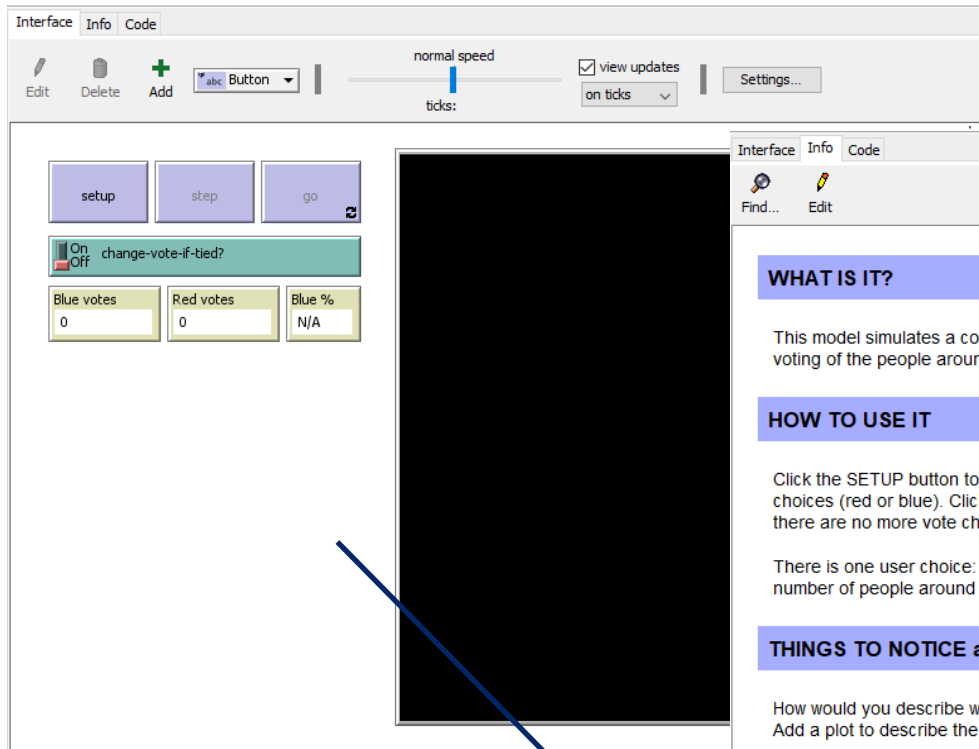
- Red darts in image
- Are mobile
- Spatially aware

Links describe relationships between turtles

- Red line in image



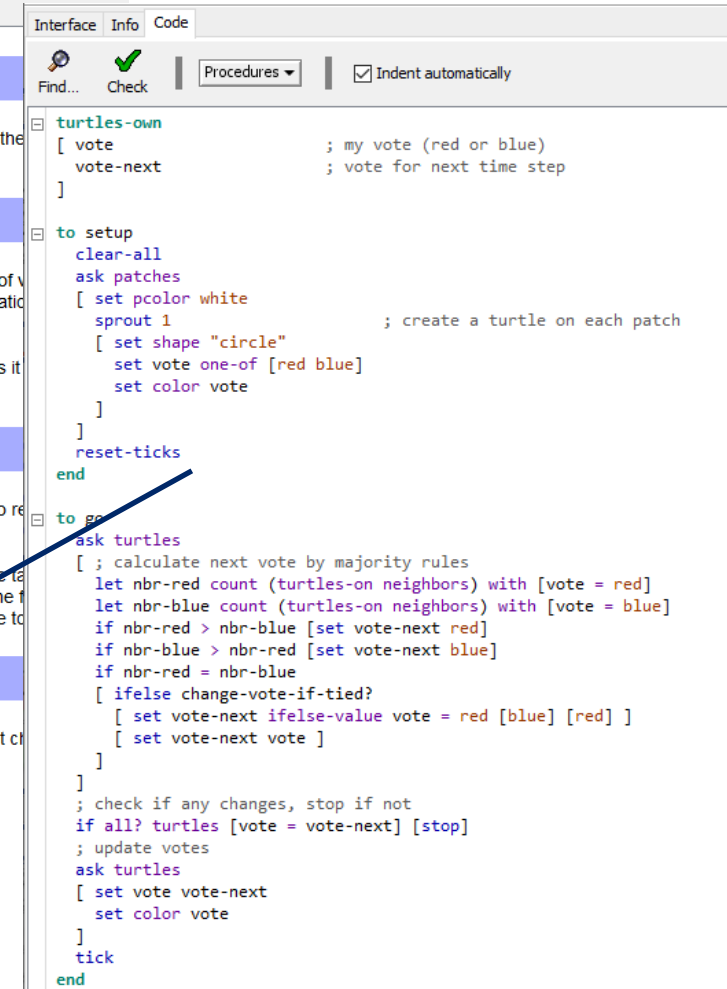
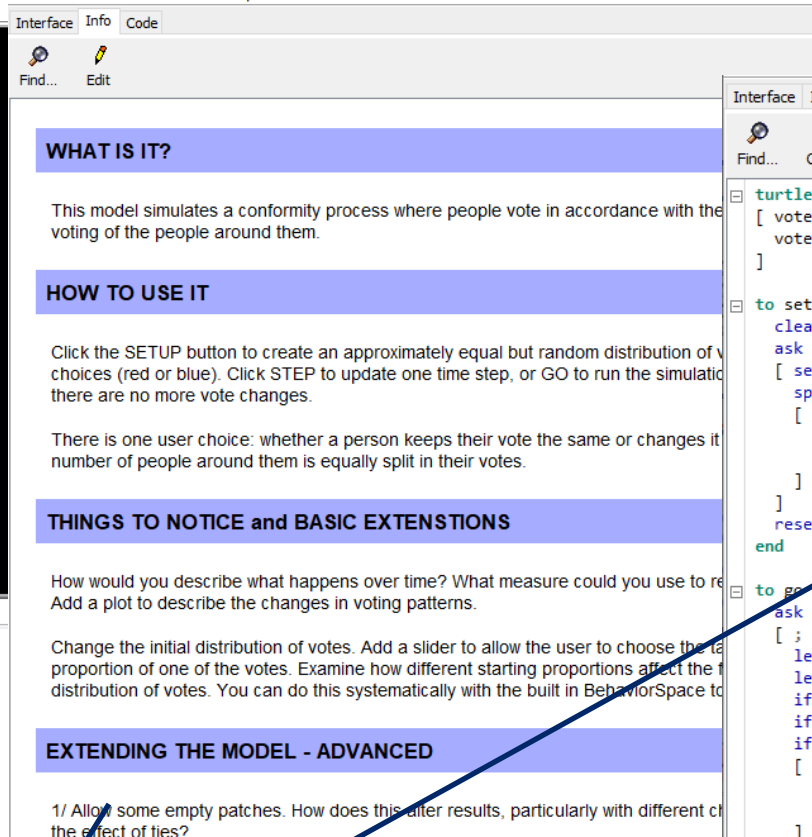
# Open Conformity model



Model interface

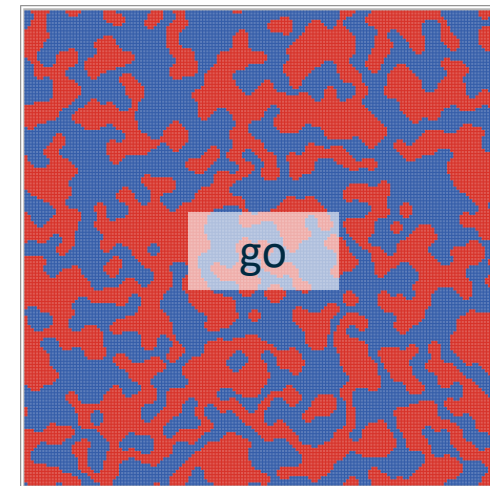
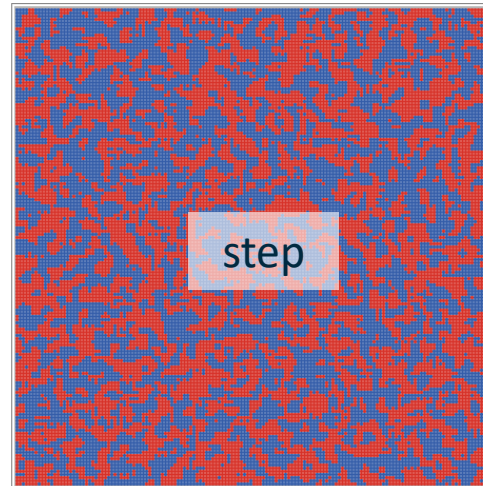
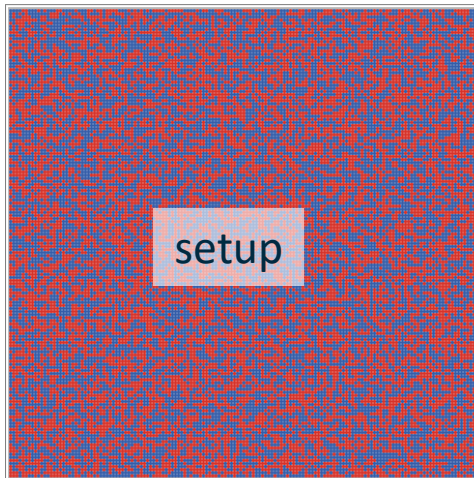
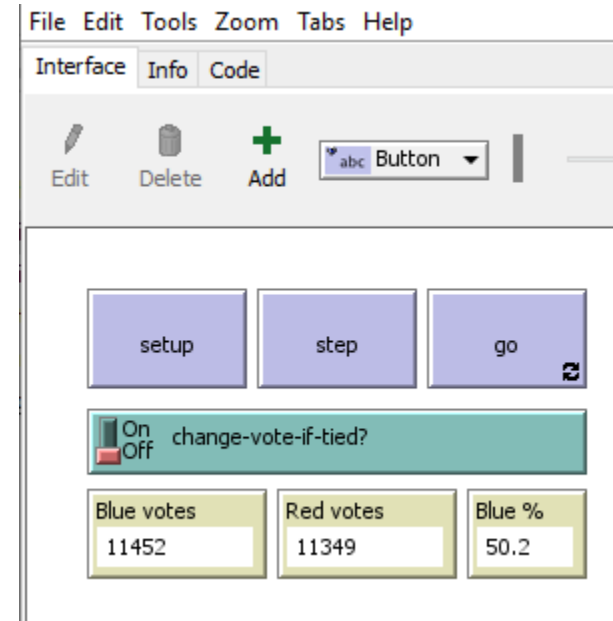
Model information  
(if modeller documents)

Code (in NetLogo language)

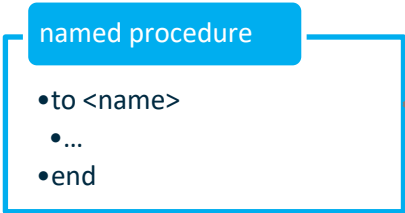
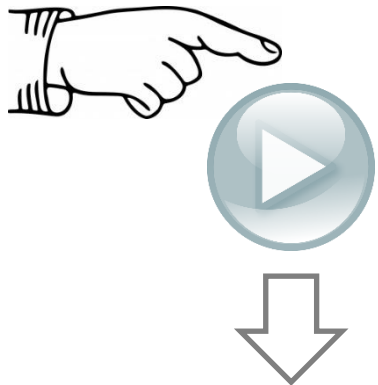


# Try it out

1. Press setup
2. Press step a few times
3. Press go
4. Repeat...
5. Change the switch
6. Discussion: what's happening?



# Modular code



Button

Agent(s)  ☐ Forever

☒ Disable until ticks start

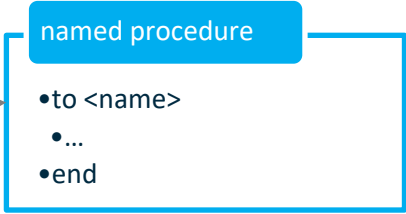
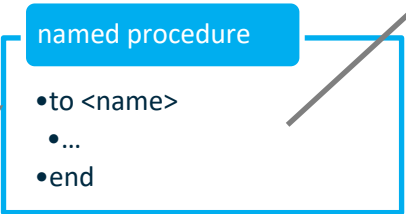
Commands

go

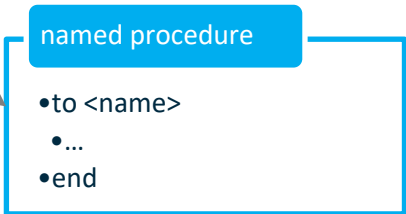
Display name

Action key

OK Cancel



...



Each procedure is made up of primitives and other procedures



# What does 'setup' do?

Defines the set of attributes available to each agent (turtle)

Deletes and clears



Each patch in random order, run the block of commands

Creates a turtle and sets attribute values

Vote variable either red or blue, equal probability

Initialises tick counter (timer)

InterfaceInfoCode

 Find... Check

Procedures ▾

☒ Indent automatically

```
turtles-own
[ vote                ; my vote (red or blue)
  vote-next           ; vote for next time step
]

to setup
  clear-all
  ask patches
  [ set pcolor white
    sprout 1                ; create a turtle on each patch
    [ set shape "circle"
      set vote one-of [red blue]
      set color vote
    ]
  ]
  reset-ticks
end
```

# What does 'go' do?

Each agent in random order, run the block of commands

Counts the red and blue voting neighbours, stores counts

Chooses an action depending on situation

Access switch on interface

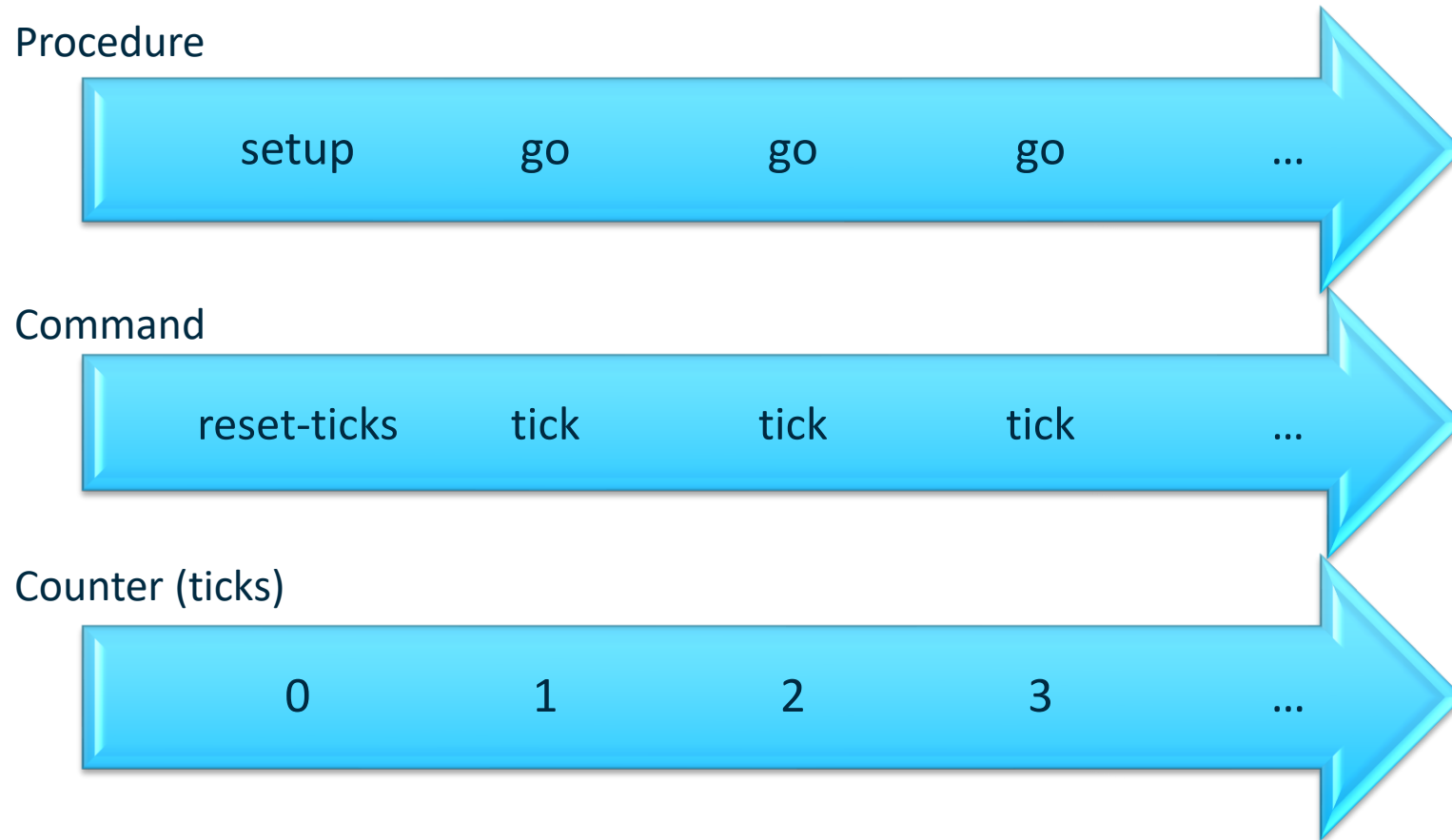
Stops if in equilibrium (all?), Cancels 'forever'

Implements chosen action

Advances tick counter (timer)

```
to go
  ask turtles
  [ ; calculate next vote by majority rules
    let nbr-red count (turtles-on neighbors) with [vote = red]
    let nbr-blue count (turtles-on neighbors) with [vote = blue]
    if nbr-red > nbr-blue [set vote-next red]
    if nbr-blue > nbr-red [set vote-next blue]
    if nbr-red = nbr-blue
      [ ifelse change-vote-if-tied?
        [ set vote-next ifelse-value vote = red [blue] [red] ]
        [ set vote-next vote ]
      ]
    ]
  ; check if any changes, stop if not
  if all? turtles [vote = vote-next] [stop]
  ; update votes
  ask turtles
  [ set vote vote-next
    set color vote
  ]
  tick
end
```

# Time in NetLogo



# How is the code 'doing' ABM?

## Simulation

- State changes: which turtle voting what
- Passage of time: tick counter

## Autonomous decisions

- Each turtle makes its own decision based on own situation

## Heterogeneity

- Some voting red, some blue

## Interaction – social

- Next vote influenced by the votes of the nearby turtles

# Extending the model: output

## What output do we have?

Current time step

World, showing vote location

Number of red / blue votes

- Calculated blue %

## What would be useful?

How would you describe the results of the model to someone who couldn't see it?

What features do you want to capture?

Small group discussion (5 mins)



## Extended output

Some measure of concentration

- Proportion blue has minimal change, but clearly the pattern changes
- Simple measure: proportion of agents that are completely surrounded by agents who vote the same way
- Formal measures available
  - Segregation index over neighbourhoods
  - Cluster sizes

Plot over time

# Making the plot

Add plot (interface widget)

- (3) in NetLogo orientation slide

Use 'select' to move or resize

The screenshot shows the 'Plot' dialog box in NetLogo. The 'Name' field is 'Surrounded by same'. The 'X axis label' is empty, 'X min' is 0, and 'X max' is 10. The 'Y axis label' is empty, 'Y min' is 0, and 'Y max' is 1. The 'Auto scale?' checkbox is checked, and the 'Show legend?' checkbox is unchecked. Below these are sections for 'Plot setup commands' and 'Plot update commands'. The 'Plot pens' section contains a table with two rows: 'red' and 'blue'. The 'red' row has the command 'plot count turtles with [color = red and all?]' and the 'blue' row has 'plot count turtles with [color = blue and all?]'.

Color	Pen name	Pen update commands
red	red	plot count turtles with [color = red and all?]
blue	blue	plot count turtles with [color = blue and all?]

click to choose colour

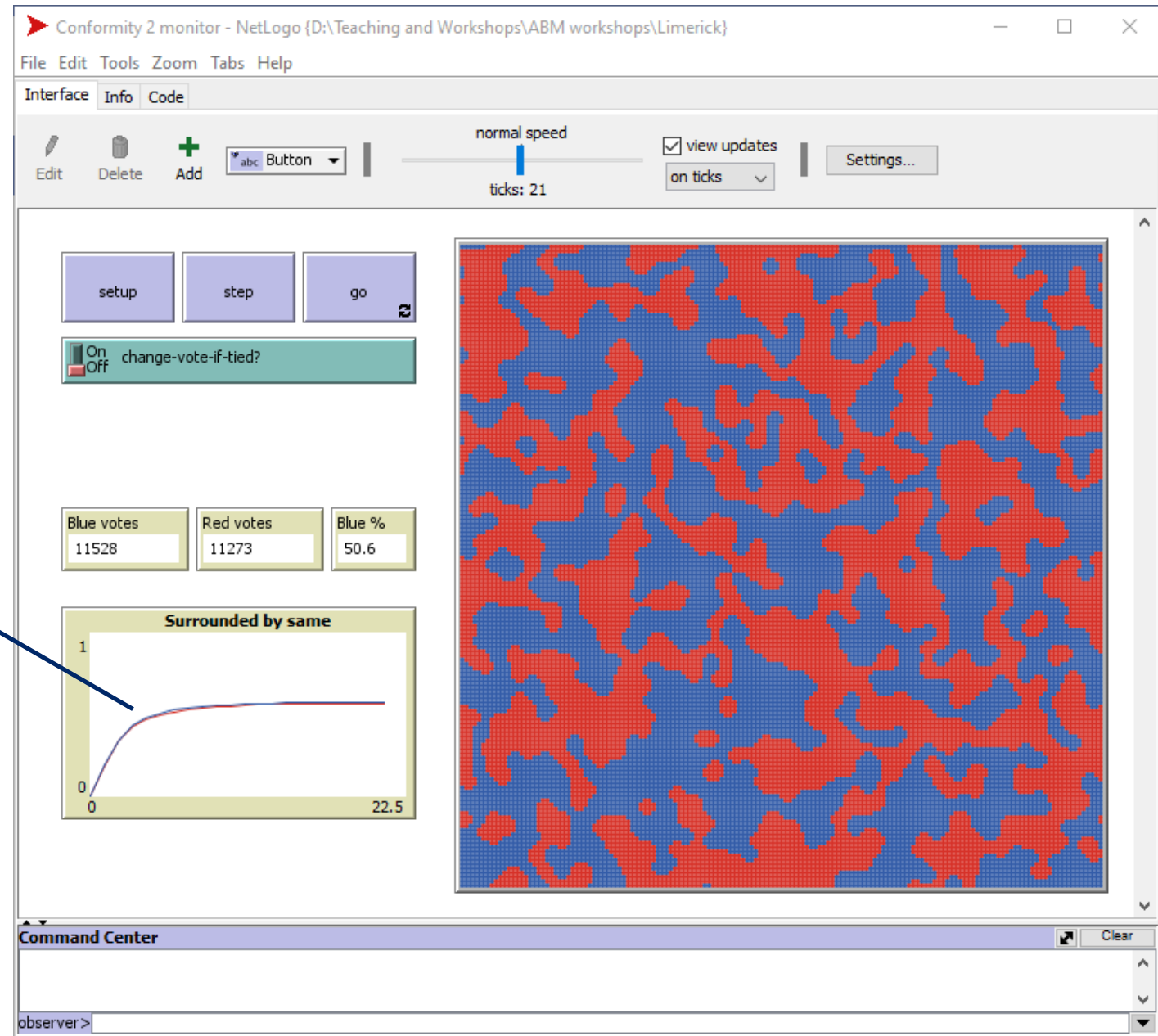
click to expand text box (and then drag border to expand further)

extra row

plot count turtles with [color = red and all? turtles-on neighbors [color = red] ] / count turtles with [color = red]

# What happens now?

Two lines



## Extending the model

### Discussion

- What could we change to ask more interesting questions?
- Small groups for ideas



# Each choice requires the mechanism to be fully specified

Start with different blue/red mix

- This is the one we will build

Heterogeneity in number of influencing agents

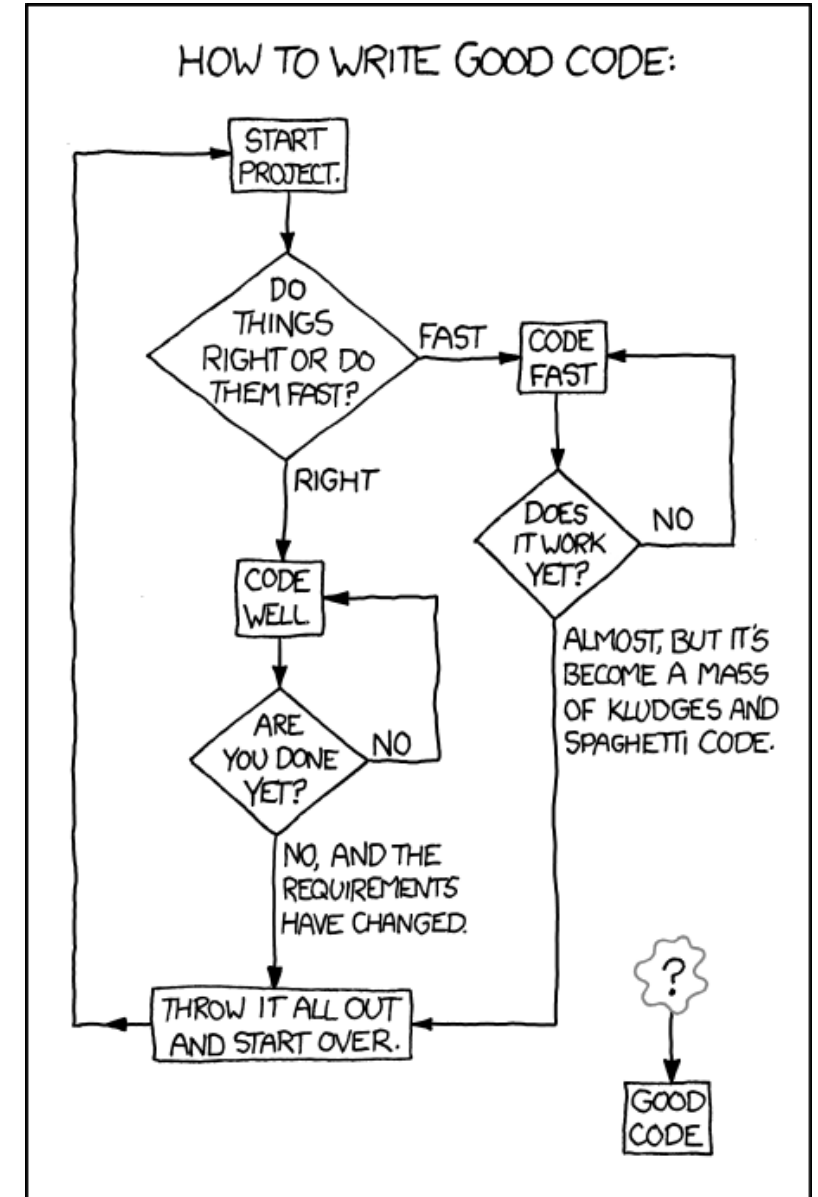
- Allow some empty patches, or more agents per patch
- Agents vary by how far they can see, not just neighbours

Heterogeneity in influence (current has equal contribution by all)

- Additional attribute of 'charisma'
- Sum of charisma rather than count of votes

Additional voting choice (colour)

- What does majority mean?

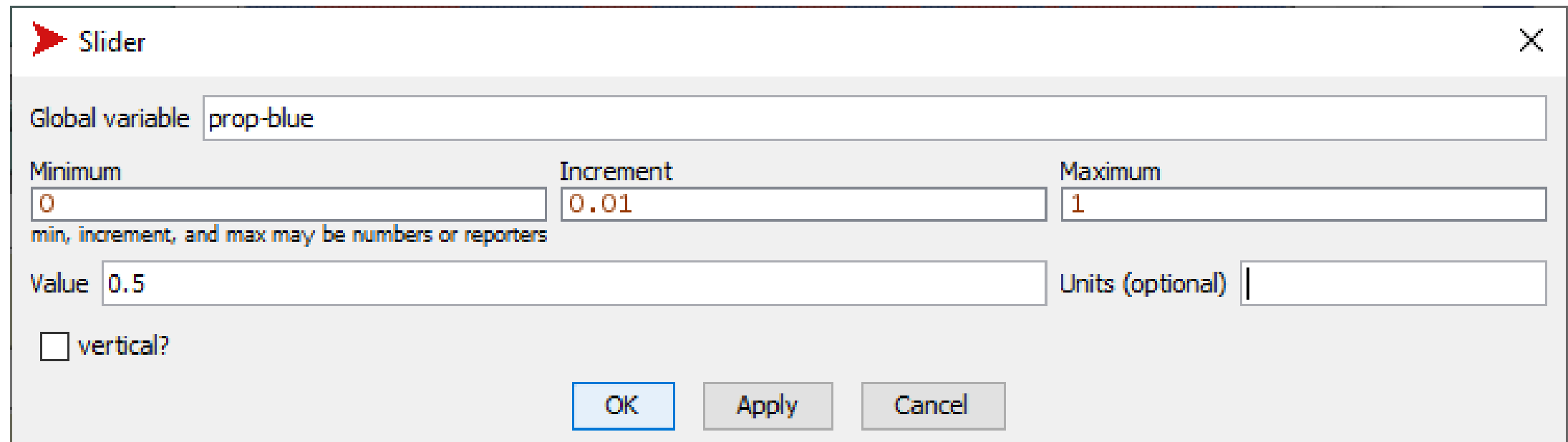


Source: Randall Munroe, xkcd comics  
[HTTP://XKCD.COM/844/](http://xkcd.com/844/)

# User control for starting distribution

Create slider for prop-blue

- Another interface widget
- Sliders allow users to set the value of some (global) variable



The image shows a 'Slider' dialog box with a red arrow icon and a close button (X) in the top right corner. The dialog contains the following fields and controls:

- Global variable:** A text field containing 'prop-blue'.
- Minimum:** A text field containing '0'.
- Increment:** A text field containing '0.01'.
- Maximum:** A text field containing '1'.
- Value:** A text field containing '0.5'.
- Units (optional):** An empty text field.
- vertical?:** A checkbox that is currently unchecked.
- Buttons:** 'OK', 'Apply', and 'Cancel' buttons at the bottom.

Below the 'Minimum' field, there is a small note: 'min, increment, and max may be numbers or reporters'.

## Using this variable in the model

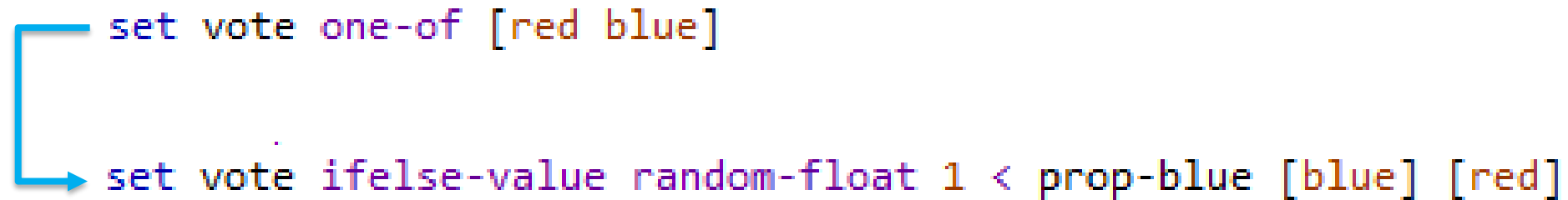
Slider makes a variable available: proportion of blue voters initially

What procedure needs to use this variable?

What code needs to change?

What does the code need to do? (Design, mechanism, logic)

- Be specific



```
set vote one-of [red blue]
```

```
set vote ifelse-value random-float 1 < prop-blue [blue] [red]
```

The diagram illustrates a change in code logic. A blue bracket on the left connects two lines of code. The top line is `set vote one-of [red blue]`. The bottom line is `set vote ifelse-value random-float 1 < prop-blue [blue] [red]`. This indicates a transition from a simple random selection to a weighted selection based on the 'prop-blue' variable.

Use the model



# Play with the model

What happens with different starting proportions?

What starting proportion is needed for both groups to exist?

How would you describe the pattern of blue proportion over time?

What is the relationship between starting proportion and:

- Final proportion?
- Final concentration?
- Do patterns change as inputs change?

Try some values!

Need more tools

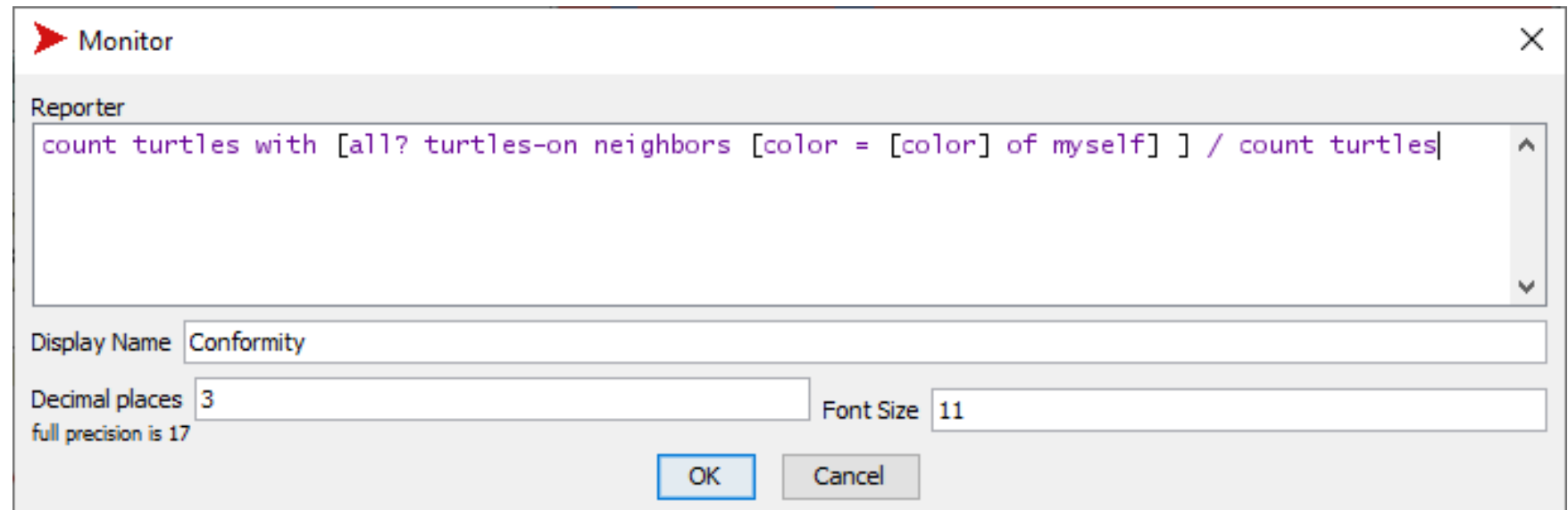


## Additional output: conformity monitor

Monitor is a widget the reports the results of a calculation

Similar code construction as the plot of surrounded

But sums the red surrounded and blue surrounded



The screenshot shows the 'Monitor' configuration window. The title bar has a red arrow icon and the text 'Monitor'. The 'Reporter' field contains the code: `count turtles with [all? turtles-on neighbors [color = [color] of myself] ] / count turtles`. The 'Display Name' field is set to 'Conformity'. The 'Decimal places' field is set to '3', with a note 'full precision is 17' below it. The 'Font Size' field is set to '11'. At the bottom are 'OK' and 'Cancel' buttons.

Monitor

Reporter

```
count turtles with [all? turtles-on neighbors [color = [color] of myself] ] / count turtles
```

Display Name: Conformity

Decimal places: 3  
full precision is 17

Font Size: 11

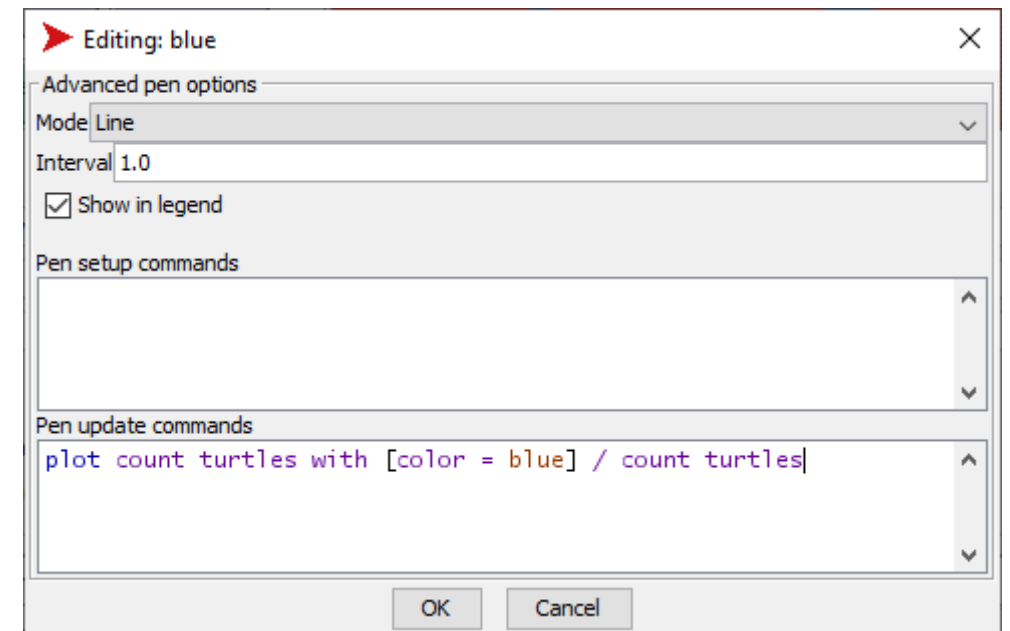
OK Cancel

# Revised plot

## Revise plot pens

- Conformity: copy the code from the monitor (and add 'plot' to front)
- Blue: make it proportion blue

Tick the legend box

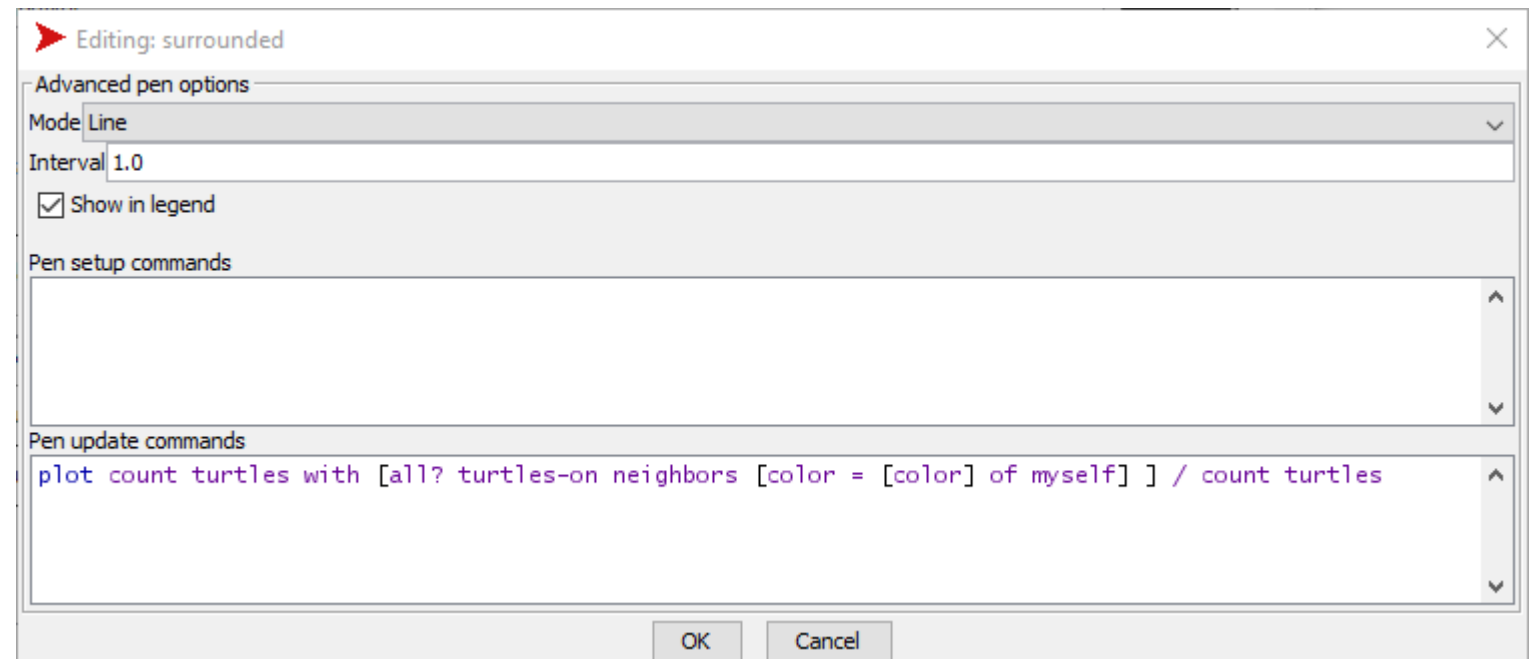


The 'Editing: blue' dialog box is shown. It has a title bar with a red arrow icon and the text 'Editing: blue'. The dialog contains the following sections:

- Advanced pen options:**
  - Mode: Line (dropdown menu)
  - Interval: 1.0 (text field)
  - ☒ Show in legend (checkbox)
- Pen setup commands:** (empty text area)
- Pen update commands:**

```
plot count turtles with [color = blue] / count turtles
```

At the bottom are 'OK' and 'Cancel' buttons.



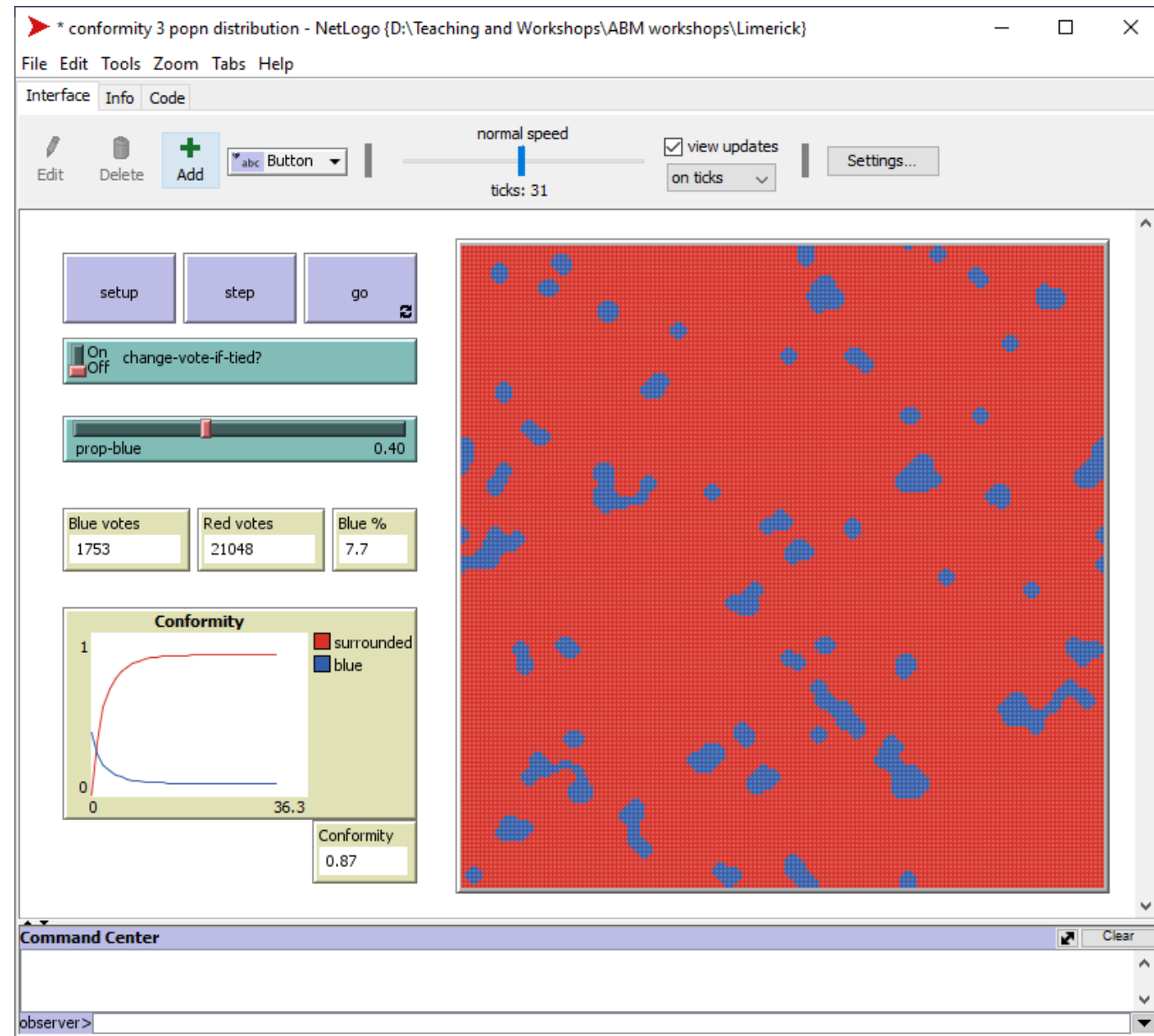
The 'Editing: surrounded' dialog box is shown. It has a title bar with a red arrow icon and the text 'Editing: surrounded'. The dialog contains the following sections:

- Advanced pen options:**
  - Mode: Line (dropdown menu)
  - Interval: 1.0 (text field)
  - ☒ Show in legend (checkbox)
- Pen setup commands:** (empty text area)
- Pen update commands:**

```
plot count turtles with [all? turtles-on neighbors [color = [color] of myself] ] / count turtles
```

At the bottom are 'OK' and 'Cancel' buttons.

# Revised model





# Experimentation

BehaviorSpace to examine

- input: prop-blue
  - fix switch to off
- output: conformity, blue

Open BehaviorSpace (Tools)

New experiment

Save (OK)

The screenshot shows the 'Experiment' dialog box in NetLogo's BehaviorSpace tool. It is used to configure a new experiment. The dialog is titled 'Experiment' and has a close button (X) in the top right corner. The 'Experiment name' field is set to 'Distribution'. Below this, there is a section for 'Vary variables as follows (note brackets and quotation marks):' with a text area containing two lines of code: `["prop-blue" [0.2 0.05 0.8]]` and `["change-vote-if-tied?" false]`. Below this, there is a section for 'Repetitions' set to '5', with a note 'run each combination this many times'. There is a checkbox for 'Run combinations in sequential order' which is unchecked. Below this, there is a section for 'Measure runs using these reporters:' with a text area containing two lines of code: `count turtles with [all? turtles-on neighbors [color = [color] ]` and `count turtles with [color = blue] / count turtles`. Below this, there is a checkbox for 'Measure runs at every step' which is unchecked, with a note 'if unchecked, runs are measured only when they are over'. There are two sections for commands: 'Setup commands' with a text area containing 'setup', and 'Go commands' with a text area containing 'go'. At the bottom, there is a 'Stop condition' section with a checkbox and a text area containing 'the run stops if this reporter becomes true', and a 'Final commands' section with a checkbox and a text area containing 'run at the end of each run'. There is also a 'Time limit' section with a text area containing '100' and a note 'stop after this many steps (0 = no limit)'. At the bottom right, there are 'OK' and 'Cancel' buttons. Blue arrows point from text labels on the right to various fields in the dialog: 'name the setup' points to the title bar, 'input sequence' points to the variable list, 'fix all others' points to the 'change-vote-if-tied?' line, '# simulations' points to the 'Repetitions' field, 'outputs' points to the reporter list, 'all or end?' points to the 'Measure runs at every step' checkbox, 'procedures' points to the 'Go commands' field, 'stop test' points to the 'Stop condition' checkbox, and 'stop tick' points to the 'Time limit' field.

Experiment

Experiment name: Distribution

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

```
["prop-blue" [0.2 0.05 0.8]]  
["change-vote-if-tied?" false]
```

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-pycor, min-pycor, random-seed.

Repetitions: 5  
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:

```
count turtles with [all? turtles-on neighbors [color = [color] ]  
count turtles with [color = blue] / count turtles
```

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: 100  
stop after this many steps (0 = no limit)

OK Cancel

name the setup

input sequence

fix all others

# simulations

outputs

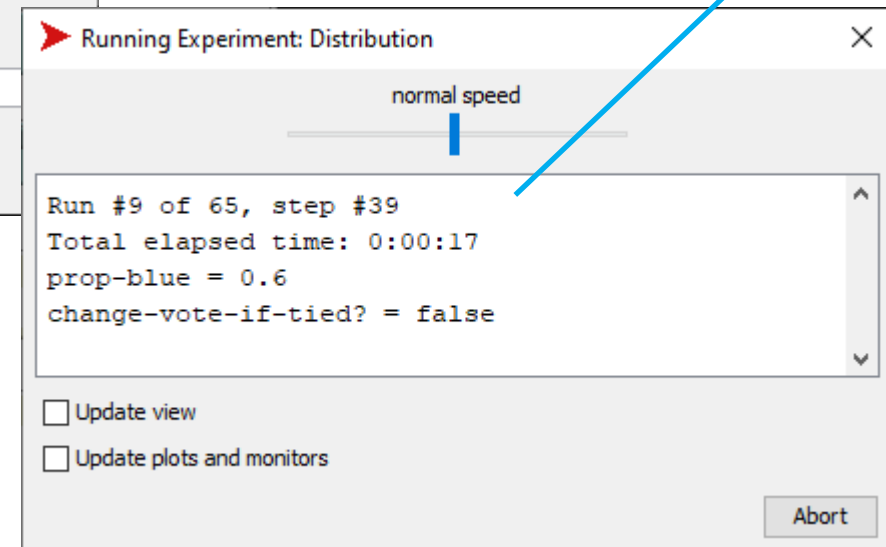
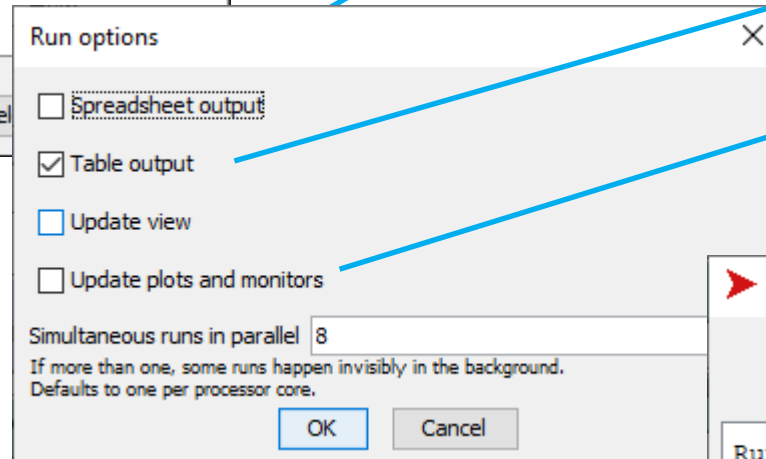
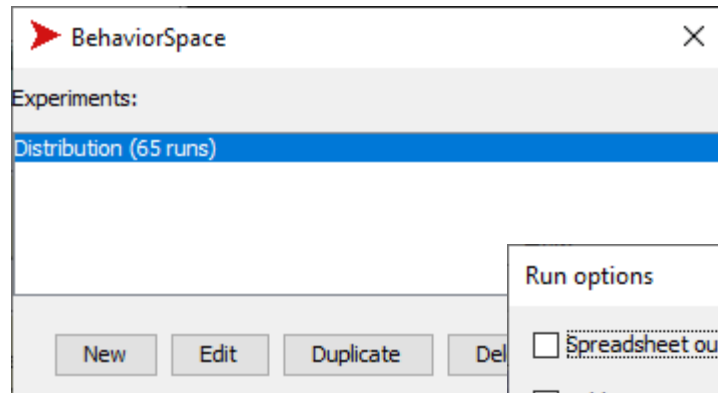
all or end?

procedures

stop test

stop tick

# Running BehaviorSpace



run button

dialogue box for file name

table output

uncheck for speed

progress

# Structure of BehaviorSpace table

	A	B	C	D	E	F	G
1	BehaviorSpace results (NetLogo 6.1.1)						
2	conformity 3 popn distribution.nlogo						
3	Distribution						
4	03/02/2020 21:31:46:604 +0000						
5	min-pxcor	max-pxco	min-pycor	max-pycor			
6	-75	75	-75	75			
7	[run num]	prop-blue	change-vc	[step]	count turt	count turtles with	
8	1	0.2	FALSE	4	1	0	
9	2	0.25	FALSE	8	0.998246	7.46E-04	
10	3	0.3	FALSE	12	0.993684	0.002675	
11	4	0.35	FALSE	12	0.950178	0.024473	
12	12	0.75	FALSE	8	0.998597	0.999474	
13	8	0.55	FALSE	25	0.685452	0.774527	
14	7	0.5	FALSE	26	0.566291	0.525635	
15	5	0.4	FALSE	25	0.841542	0.088417	
16	13	0.8	FALSE	4	1	1	
17	11	0.7	FALSE	15	0.992456	0.99693	
18	6	0.45	FALSE	31	0.645849	0.264111	
19	10	0.65	FALSE	20	0.962633	0.982983	
20	14	0.2	FALSE	6	1	0	
21	15	0.25	FALSE	7	1	0	
22	16	0.3	FALSE	9	0.989474	0.004386	

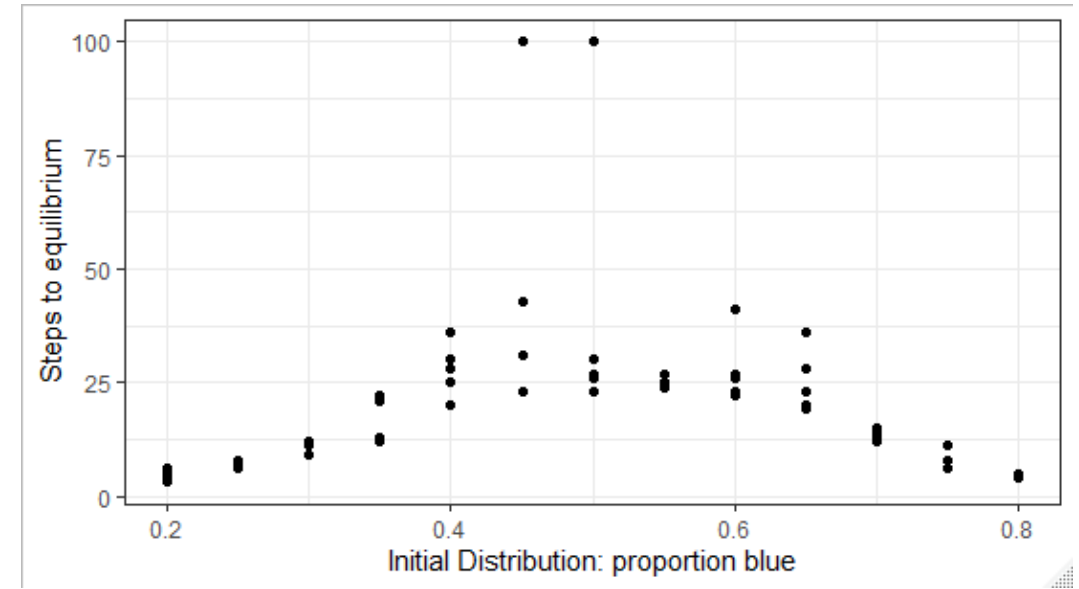
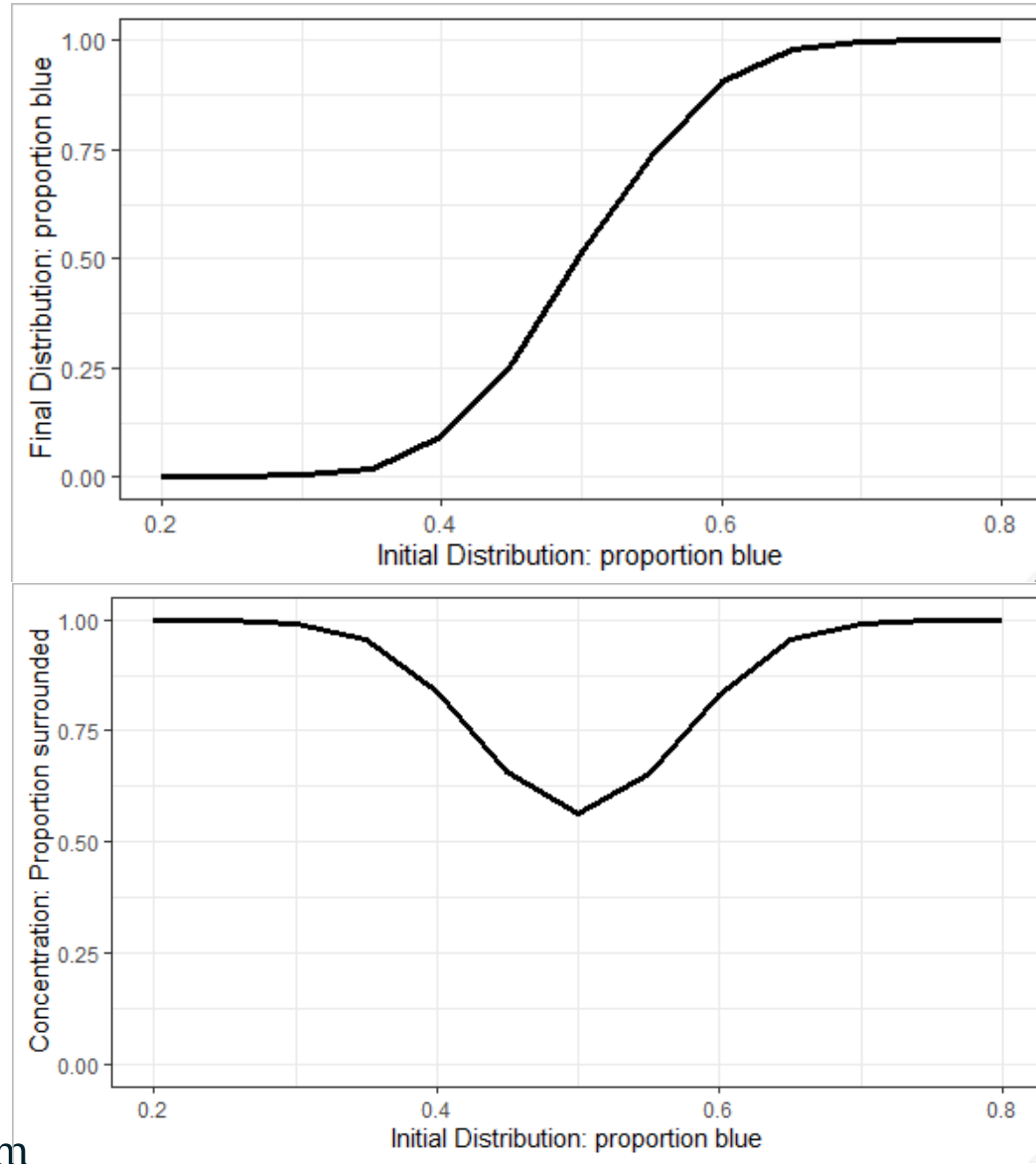
## Top

- 4 lines file info
- 2 lines world size

## Body

- column with run number
- column(s) for parameter values
- column for step number
  - In this case, gives ticks until equilibrium reached
- column(s) for reporters

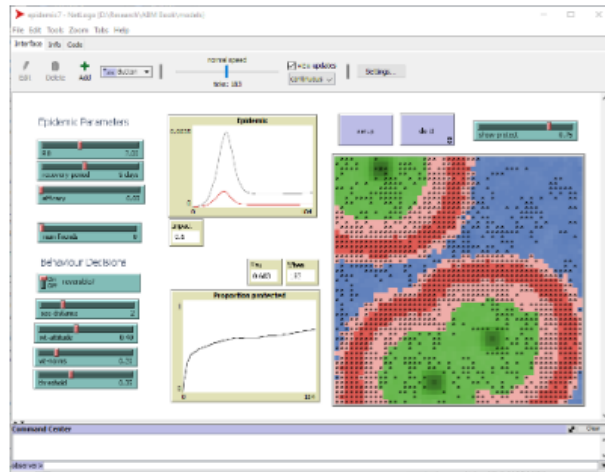
# Analyse results



# If you want to learn NetLogo...

<https://jbadham.biz/Research/ABMBook/>

## Agent-Based Modelling for the Self Learner



### What is this tutorial?

Agent-based modelling is a complex systems method to simulate individuals making decisions based on their own characteristics, social influences and situation. For the past few years, colleagues and I have been running ABM short courses. Those courses include a substantial tutorial in NetLogo, freely available specialist software.

Many people wanting to use agent-based modelling are sociologists, public health researchers, geographers, environmental scientists or other disciplinary based scientists who may not have programming experience or access to experienced agent-based modellers. I have adapted the tutorial from the course so that such people can work through building a model of protective behaviour during an influenza epidemic. As well as the Netlogo language and programming

environment, the tutorial is intended to teach the way that agent-based models represent the world and good programming practices. At some point, I hope to extend the tutorial into a full textbook.

I hope you find it useful. Comments and suggestions are welcome, by [email](#).

*Jen Badham* (June 2019)

### What do you need?

NetLogo software is freely available for Windows, Mac OSX or Linux, from the developers at: [Netlogo](#).

Download the (pdf) tutorial [here](#). It assumes no background in programming or agent-based modelling but you are expected to be comfortable with standard computer operations such as saving a file.

Some of the models that you will build are also available:

- Progress versions of the main model ([model 1](#), [model 2](#), [model 3](#), [model 4](#)) and the [completed](#) model.
- The additional tutorial for mobile agents is only available in [completed](#) form.

# Key Ideas



# Modelling a complex system

## Concepts

Complex system: Behaviour arises significantly from interactions

Model: Simplified representation of features and relationships

Simulation

- Model of a process
- States change over time

Simulation of a complex system describes some theory about state changes that involves interactions

## When to use ABM

ABM only one method for modelling complex systems

ABM used where:

- Autonomous individuals (agents)
  - No central control
- Agents interact with each other and/or their environment
  - Local effects
- Agents are diverse
  - Heterogeneity contributes to different actions

# Why NetLogo for ABM?

## Easy to learn

- No other language
- Small command set for most requirements

## Models are quick to write

- Integrated interface and code
- Few lines of code

## Sufficiently powerful

- Suitable for up to about 50,000 agents

## Clunky graphics

- Reminds users that they are working with a model, not a predictive tool





## Models assist insight



Tool for thinking: Agent-based model captures some theory of process and allows us to examine the implications of that theory