GEOG3150 SEMSESTER 2

LECTURE 2

USING NETLOGO FOR INDIVIDUAL-LEVEL MODELLING

Dr Nick Malleson
Dr Alison Heppenstall
Dr Nik Lomax

Click here for full screen

AIMS OF THE LECTURE

This lecture will introduce the NetLogo modelling tool and explain the basic concepts. By the end of the lecture you will be familiar with the basic elements of NetLogo and be able to start using it to build models!

How to use these slides

These slides are made using html, so they need to be read on-line. You can use the arrows in the bottom-right corner to move between slides, or press the right/left arrows on your keyboard. Pressing escape gives an overview of all slides.

There are also notes for some of the slides. To see these, either print out the slides (instructions below) or press the 's' key. This puts you into a different mode that will show notes alongside slides.

If you would like to print them out for offline reading, or save them as a pdf, you need to add '`?print-pdf`' to the end of the url, like so:

http://www.geog.leeds.ac.uk/courses/level3/geog3150/lectures/lecture2/lecture2.html?print-pdf

Then you can print as normal (e.g. File -> Print). Depending on the version of your browser, you might also need to select'landscape' paper type..

Important: printing only works using Google Chrome

Caveat

This lecture is quite dry, sorry!

But at least this stuff wont come up in the exam...

Programming ...

In the next few weeks you will learn how to write computer code.

- Don't panic!
- The tasks are easy, but computers are stupid
- NetLogo was designed for children - a nice introduction to programming.

Other benefits

- It will change the way you think (honestly, it will)
- An incredibly valuable CV addition
- Open new, exciting opportunities to research the world

If you are inspired to learn more...

- Try: **Code Academy** (and their **python tutorials**).
- And: **code.org**.

Computer programming is set to become a key skill that all students will learn from an early age in order to participate in a world that is becoming increasingly digital. For geographers, being able to write write computer code opens up enormous possibilities for new and exciting areas of research. Huge volumes of spatial data are becoming available that geographers can exploit to learn about the world. Being able to program allows you to escape the restrictions that traditional geographic software and services (such as Google Maps) place on the types of analysis you can do.

It does, of course, require much more than this single course to become a proficient programmer, capable of harvesting and exploiting new data to discover new things about the world. However, by learning to build some simple but powerful computer models using NetLogo you will have a basic grasp of how computer programs work - the next level of understanding is not much more than a small step away.

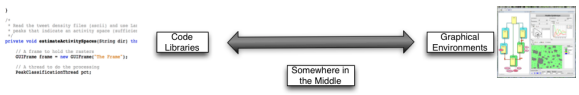
*If you are keen to learn more about programming, there are great online resources. Have a look at **Code Academy** and, in particular, their **python tutorials** (python is an increasingly popular computer language).*

Don't panic! At times you will find NetLogo programming frustrating, but there is lots of help available.

Outline

1. Tools for individual-level modelling
2. **Introduction to NetLogo**
3. **The Program**
4. **Turtles and Patches**
5. **Variables**
6. **Flow Control**
7. **Writing Nice Code**
8. **Summary**

Software Tools / Platforms



What are they?

- Pieces of software to help people build models

Wide range of tools

- Computer code ('libraries')
- Entire graphical environment
- Somewhere in the middle
- ... and somewhere else ...

Computer code ('libraries')

Researchers write software to perform useful functions:

- Draw graphs
- Visualise the model
- Manage the schedule

Great for programmers

- Less time spend worrying about admin, more time on modelling

Examples:

- **MASON**
- **Repast Simphony**
- **Mageo**
- Loads of others listed **here**

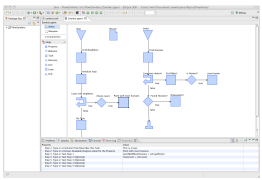
Graphical Environments

Entirely visual - no programming needed

Most useful for non-programmers

Examples

- **Agent Sheets**
- **VisualBots**
- **Repast Simphony**
- **Modelling4All**



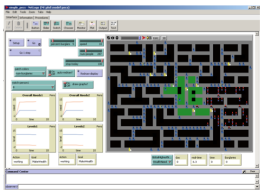
Somewhere in the middle

Some code writing, some visual development

More powerful than purely visual tools, but easier to use.

Save time having to learn to do simple tasks and concentrate on model behaviour

e.g. NetLogo



… somewhere else …

There are loads of other packages that people are using in novel ways. E.g.

Second Life (see **http://www.casa.ucl.ac.uk/abm/secondlife/**

Outline

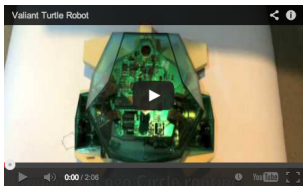
1. **Tools for individual-level modelling**
2. **Introduction to NetLogo**
3. **The Program**
4. **Turtles and Patches**
5. **Variables**
6. **Flow Control**
7. **Writing Nice Code**
8. **Summary**

Base on **Star Logo**.

Popular teaching tool

Designed to be used by children

But also powerful



Developed by The Center for Connected Learning (CCL) and Computer-Based Modeling at Northwestern University

Free!

Uses Java in the background

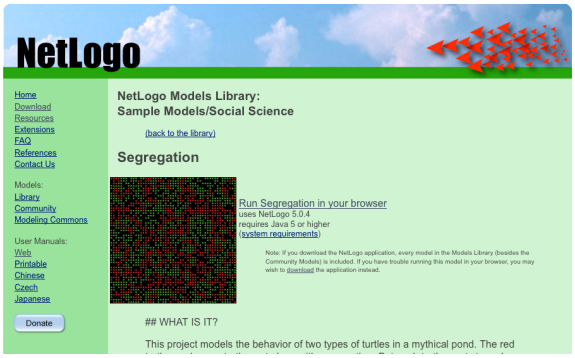
- **Multi platform**
- Can be converted into applets (and embedded in websites)

Great for quickly putting a model together and thinking through ideas

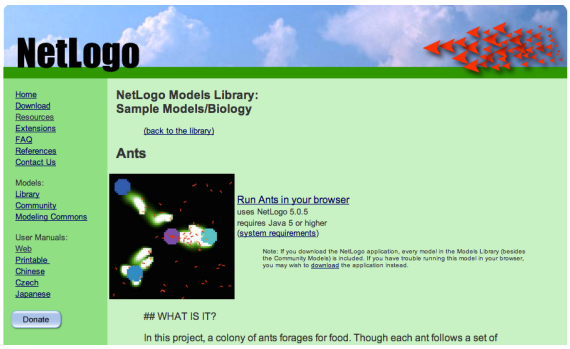
- Easy to build
- Easy to interact with models
- East to extract data and create plots

Excellent documentation:
http://ccl.northwestern.edu/netlogo/docs/

Example - Segregation (Schelling)



Example - Ants



Outline

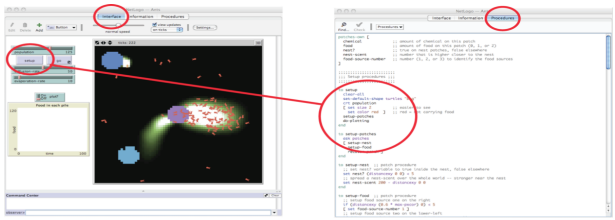
1. **Tools for individual-level modelling**
2. **Introduction to NetLogo**
3. **The Program**
4. **Turtles and Patches**
5. **Variables**
6. **Flow Control**
7. **Writing Nice Code**
8. **Summary**

The Program

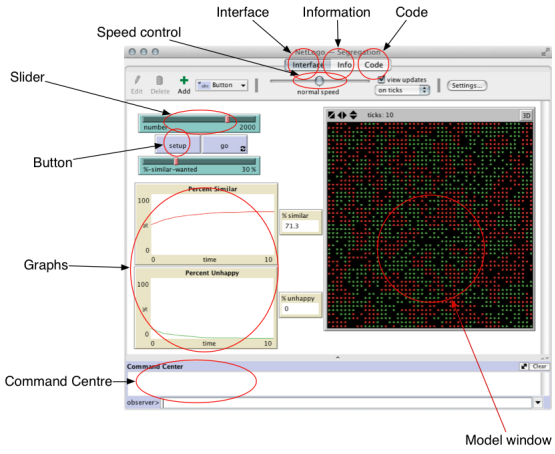
NetLogo is "somewhere in the middle"

Graphical part (**Interface**) with sliders, graphs, buttons and a map

Scripting part (**Procedures**) which contains instructions (code)



The Interface



Interface Components

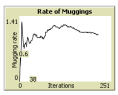
Switch

Slider

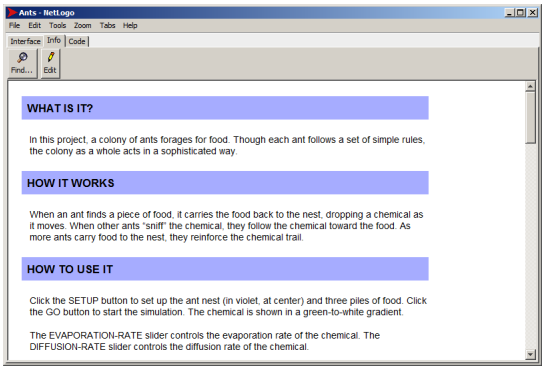
Button

Monitor

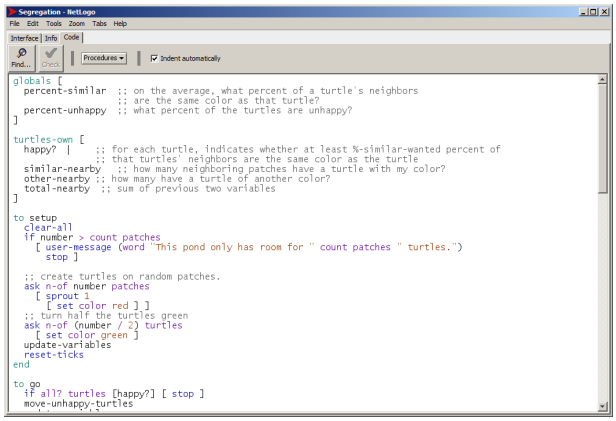
Graph



The Information Tab



The Program - Code



Outline

1. **Tools for individual-level modelling**
2. **Introduction to NetLogo**
3. **The Program**
4. **Turtles and Patches**
5. **Variables**
6. **Flow Control**
7. **Writing Nice Code**
8. **Summary**

Turtles, Patches and the Observer

There are two types of objects in NetLogo: **turtles** and **patches**.

Both are *agents*

- They have rules that determine their behaviour
- They can interact with other agents

Main differences:

- Patches cannot move
- You can create different types of 'turtle' (e.g. person, dog, cat, car, etc.)

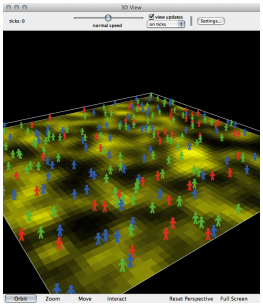
Why turtles?

'Logo' language originally used to control robot turtles. It seems that the name 'turtle' has stuck..

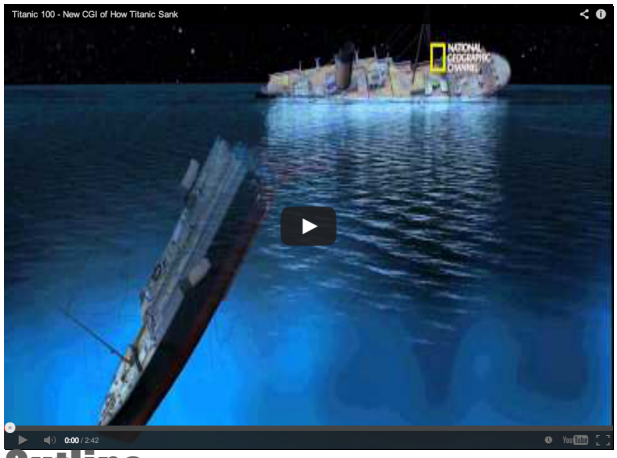
Turtles, Patches and the Observer

Also important: the **observer**

- The 'god' of a model
- Oversea everything that happens, give orders to turtles or patches, control other things like data input/output, virtual time, etc.



... short break ...



Outline

1. **Tools for individual-level modelling**
2. **Introduction to NetLogo**
3. **The Program**
4. **Turtles and Patches**
5. Variables
6. **Flow Control**
7. **Writing Nice Code**
8. **Summary**

Variables

In programming, variables are a way of storing information. E.g.

```
my-name = "Nick"
seconds-per-minute = 60
pi = 3.142
infected = True
```

Variables can *belong* to different objects in the model.

Examples:

- **Turtle variables**: e.g. `name`, `age`, `occupation`, `wealth`, `energy`
- **Patch variables**: e.g. `height-above-sea`, `amount-of-grain`, `building-security`, `deprivation`
- **Observer variables**:`total-wealth`, `weather`, `time-of-daypi`

Different objects can have different *variable values*

Which variables?

Scenario: You are building a model of car traffic and need to decide how to implement the behaviour of your drivers.

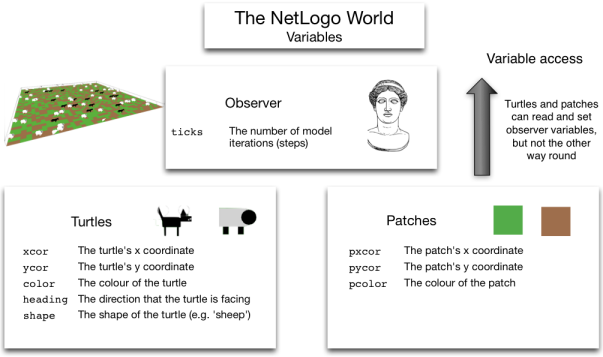
What are the most important things that will determine how a driver reacts in a certain situation?

This is a video produced by the ***PTV Vissim***

These will determine the **variables** that need to be included in a model

Current Speed
Driver's Age

Built-In Variables in NetLogo



NetLogo Commands

Commands are the way of telling NetLogo what we want it to do

Some examples

(don't worry, these will be explained properly in the **first practical**):

`show "Hello World"`	Prints something to the screen
`set my-age 13`	Sets the value of a variable
`ask turtles [ ... ]`	Ask the turtles to do something
`ask turtles [ set color blue ]`	Asks the turtles to turn blue

Commands are very **well documented**

Brackets

NetLogo uses both square `[ ]` and round `( )` brackets.

Round brackets are used to set the *order of operations*. E.g.:

```
 2 + 3  × 4 = 14
(2 + 3) × 4 = 20
```

Square brackets are used to split up commands. E.g.:

```
ask turtles [ ... ]
```

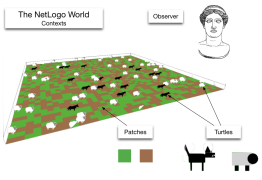
the `ask` command expects to find some more commands inside the brackets.

Contexts

Contexts are NetLogo's way of controlling where commands are sent.

There are three contexts:

1. Observer
2. Turtle
3. Patch



Don't Panic: Lots of opportunity to understand these during the practicals..

Outline

1. **Tools for individual-level modelling**
2. **Introduction to NetLogo**
3. **The Program**
4. **Turtles and Patches**
5. **Variables**
6. **Flow Control**
7. **Writing Nice Code**
8. **Summary**

Flow Control

Programs are recipes

And computers are really, really stupid cooks.

Programmers need to tell the computer *exactly* what to do, and in what order

Geek joke:

> Q: How do you keep a programmer in the shower forever?
> A: Give him a bottle of shampoo that says "lather, rinse, repeat".

Flow Control and Logic

Usually, NetLogo will run through your code, one line after the other.

But! Sometimes there are two or more possibilities for what to do next.

`if` statements are one example:

```
... start here ...

if ( age < 18 )
  [ .. do something .. ]

if ( age > 18 )
  [ .. do something else .. ]

... now continue ...
```

Flow Control Quiz

The code below has been taken from the rules that drive the behaviour of a virtual person (or 'agent'). What will the person do when the `age` variable has these different values:

Age	10	50	18
Actions	?	?	?

```
... start here ...

if ( age < 18 )
  [ .. go to the cinema .. ]

if ( age > 18 )
  [ .. go to the pub .. ]

.. go to my friend's house ..

... now continue ...
```

Outline

1. **Tools for individual-level modelling**
2. **Introduction to NetLogo**
3. **The Program**
4. **Turtles and Patches**
5. **Variables**
6. **Flow Control**
7. **Writing Nice Code**
8. **Summary**

Finally: Writing Nice Code

Computers don't care what code looks like

But there are some good conventions that we can use to make our code easier to understand by humans

Indentation

- New blocks of code should be *indented* (moved to the right)
- E.g. the `if` statements on previous slide

White space

- Different sections of code can be separated by lots of white space

Comments

- Comments are special parts of code that NetLogo will ignore.
- Anything after a `;` is ignored.
- Use comments to explain what your computer code does.

Indentation

Good

```
if age = 15 [

  if count friends > 0 [
    set happiness ( happiness + 1 )
  ]

  if count friends > 5 [
    set happiness ( happiness + 5 )
  ]

]
```

Bad

```
if age = 15 [

if count friends > 0 [
set happiness ( happiness + 1 )
]

if count friends > 5 [
set happiness ( happiness + 5 )
]

]
```

Whitespace

Good

```
if age = 15 [

  if count friends > 0 [
    set happiness ( happiness + 1 )
  ]

  if count friends > 5 [
    set happiness ( happiness + 5 )
  ]

]
```

Bad (well, not too bad, but ..)

```
if age = 15 [
  if count friends > 0 [
    set happiness ( happiness + 1 )
  ]
  if count friends > 5 [
    set happiness ( happiness + 5 )
  ]
]
```

Comments

Good

```
if age = 15 [

   ; This happens if the agent is 15 years old
  if count friends > 0 [
   ; If at least 1 friend, then they're happy
    set happiness ( happiness + 1 )
  ]

  if count friends > 5 [
   ; If they have 5, then even more happy
    set happiness ( happiness + 5 )
  ]

]
```

Bad

```
if age = 15 [

  if count friends > 0 [
    set happiness ( happiness + 1 )
  ]

  if count friends > 5 [
    set happiness ( happiness + 5 )
  ]

]
```

Summary

1. **Tools for individual-level modelling**
2. **Introduction to NetLogo**
3. **The Program**
4. **Turtles and Patches**
5. **Variables**
6. **Flow Control**
7. **Writing Nice Code**

Next Week

Social Simulation

Introduction to Agent-based Models

Seminar 1: **GIS and Geocomputation**

Next Week's Seminar

Seminar 1 - GIS and Geocomputation

Compare and contrast Geo-computation methods with the GIS approach.

Details:
http://www.geog.leeds.ac.uk/courses/level3/geog3150/seminars/seminar1/

Reading

First half of chapter 2 from Gilbert and Troitzsch (2005).

> *Gilbert, Nigel and Klaus G. Troitzsch (2005) Simulation for the Social Scientist. Open University Press*

Then this one page opinion piece in *Nature*:

> *Epstein, J.M., (2009) Modelling to contain pandemics. Nature 460, 687-687*

.