Introduction to NetLogo and Python

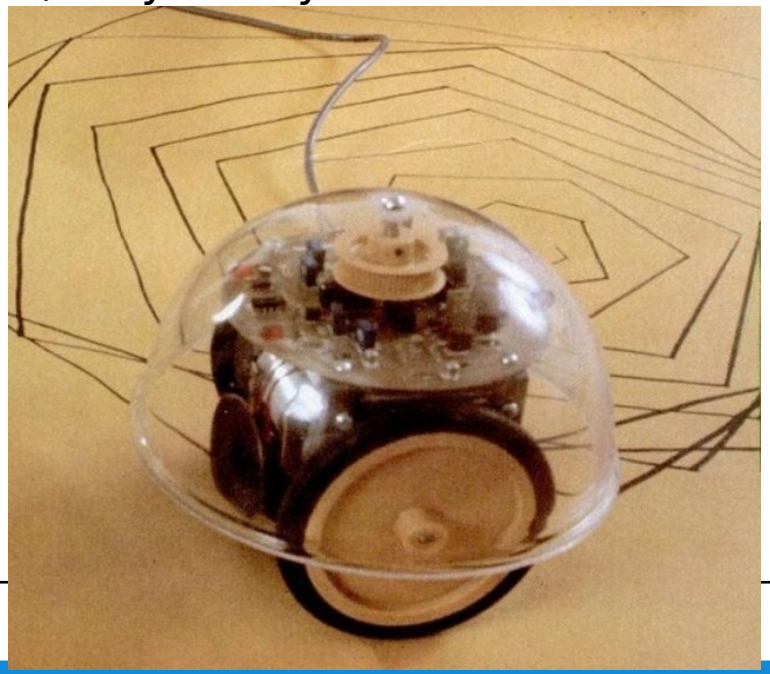
Dr. ir. Igor Nikolic
11-11-21

Lecture goals

- A guided tour of NetLogo and mesa
- How to learn the tools

Netlogo

- Open source and free language and development environment for Agent based Models
- Defacto standard in computational social science, very widely used in teaching
- https://ccl.northwestern.edu/netlogo/
- inspired by Logo language
- "low threshold and no ceiling"
- open source and free
  - https://github.com/NetLogo/NetLogo
- written in Scala, runs in a JVM



Anatomy of a basic ABM in NetLogo

NetLogo - useful features (to read later as you are learning)

- turtle/patch/links/observer - breeds
- globals vs turtles-own vs globals in gui
- to vs to-report
- let vs set
- ask
- List / arrays
- map, reduce, foreach
- matrices and tables
- me vs myself

turtle/patch/links/observer - breeds

- In Netlogo we have 4 types of agents :
  - turtles, patches, links, and the observer.

- Turtles are “normal” agents (network nodes)
- Patches cant move
  - Fixed position in the world
  - Middle of the screen is 0,0
  - Check whether the x,y wrap is enabled or not
- Links are special types of turtles that connect other turtles
- Observer is the model itself (i.e. world/simulation/you)

- Breeds allow you to differentiate between agent types
  - breed [wolves wolf]
  - breed [sheep a-sheep]

globals vs turtles-own vs globals in gui

- Globals
  - Variables that are accessible to all
  - State of the world/environment
  - ! Can also be defined in the GUI ! :(
- Turtles own
  - Turtle states
  - Another turtle can **ask** for them
- Breeds
  - Next to turtles own, that all turtles have, you can have breeds own, that only that breed has

to vs to [] vs to-report

- to doAThing
  - Functions that do something
    - **to** .... **end**
- to doAThing [*somethingElse*]
  - Function that does something with *somethingElse*
- to-report aThing
  - Functions that return a value
  - To-report …
  - report
  - end

let vs set

- **Let**
  - Defines (and possibly sets) a variable in the **current scope**
    - Inside **to … end** and/or **[…]**
- **Set**
  - Sets the value of a variable
- For both, unusual syntax :
  - **set/let** VARIABLE VALUE

ask

- Iterates over an **agentset**
  - **Turtles/links** (all turtles/all links)
  - **breeds** (all turtles of that breed)
  - Always randomizes the order of iteration
  - You can also ask a single agent
  - Ask turtle x [….]
- You can nest asks
  - Be careful about
    - ask turtles [ask turles […]]
- Always keep in mind who is asking (context)

Lists

- Very convenient
- Can cheaply change length
- A string “text” is also a list
- foreach command / map reporter are useful, learn them
- List **cant not** be modified, only overwritten
  - set mylist lput 42 mylist
  - Carefully read about the list primitives !

map

- show map round [1.1 2.2 2.7]
  => [1 2 3]

- show map [i -> i * i] [1 2 3]
  => [1 4 9]

- show (
  map [[a b c] -> a + b = c]
  [1 2 3] [2 4 6] [3 5 9]
)
  => [true false true]

reduce

- show reduce + [1 2 3]
  => 6
- show reduce - [1 2 3]
  => -4
- show reduce [[ignored next-item] -> next-item] [1 2 3]
  => 3
- show reduce [[result-so-far ignored-item] -> result-so-far] [1 2 3]
  => 1

foreach

- foreach [1.1 2.2 2.6] show
  => 1.1
  => 2.2
  => 2.6

- foreach [1 2 3] [2 4 6]
    [[a b] -> show word "the sum is: " (a + b)])
  => "the sum is: 3"
  => "the sum is: 6"
  => "the sum is: 9"

Matrix / table / network extensions

- extensions
- have their own primitives
- don’t play that well with other netlogo features
- use if you really have to

Self vs myself

- **Observer** does this
- Ask turtles [set xcor 7]
  - Puny turtles! I am the Observer! Obey me! Set YOUR xcor to 7!

- ask turtles with [self != myself]
- [rt who, fd who-of myself]
  - Foolish turtles! I am turtle 15! Obey me! If you (self) are not me
  - (myself), you must turn right a number of degrees equal to YOUR who number (who, implies who-of self), then move forward a number of units equal to MY who number (who-of myself)! I am finished with you!

- To go ; observer does this:
- ask patches with [self != patch 0 0]
- [set pcolor blue]
- End

- Insignificant patches! I am the Observer! Obey me! If YOU are not patch 0 0, turn YOUR pcolor blue! That is all!

- To go *; observer*
- ask patches with [any? turtles-here] *; observer is doing the asking!*
  - [ask turtles-here *; patches are doing the asking!*
    - [set color red, jump 2, set pcolor blue]
  - set pcolor white] ; observer does the asking

- Observer: Unworthy patches! I am the observer! Obey me! Do this ask clause!
- Patches: Pathetic turtles-here! I am the patch under you! Obey me! Change color to red! Jump 2 units! Change the pcolor (of the new patch under you) to blue! That is all!

- Observer: Now, insolent patch! Change your pcolor to white! I am finished with you!

mesa / python

- Open source and free Library for python
  - “goal is to be the Python 3-based counterpart to NetLogo, Repast, or MASON.”
- Relatively new, current version 0.8.7
- https://github.com/projectmesa/mesa

Anatomy of a basic ABM in mesa

How to learn tools

- Read the manual / API at least once
  - netlogo programing guide
  - nelogo dictionary !
  - netlogo example modes. DO NOT copy paste, type them in.
- **Understand** example models on brightspace
  - this means, understand every single thing in the code
    - yes, it takes forever the first time
    - google / ask what you do not understand
- Write out / draw out the logic of the complete model and agent behavior (flowcharts)
- Think about the change, while looking at the drawing
- Implement the change
- Run/test