Introduction to Agent-Modeling with Netlogo

Alexander Elkholy, Daniel Korytow
Agent based modeling allows us to focus on individual agents and their interactions.

- Agents are given local (micro) rules and behaviors for interaction.
- These interactions can give rise to complex patterns and phenomena (macro).
Emergent behavior is a macro level phenomenon which arises from the simple, low level rules governing agents. For example:

- Birds will flock together and move in unity, even if no single bird is in control.
Stock markets are influenced by herd behavior, where individual traders do not have a significant effect on the market, but large numbers can.

Traffic jams can emerge from many cars interacting even without a collision, just from individuals preferring different speeds.
This group of locally interacting agents giving rise to emergent behavior is called a Complex Adaptive System.

These systems are determined by what is called a "bottom up process"
Netlogo defines four objects:

- **Turtles**: Agents which move about in the world (two dimensional, divided into patches).
- **Patches**: The grid/ground the Turtles live on.
- **Links**: Connections between turtles.
- **Observer**: Gives instructions and can create other agents.
The way to give commands to turtles is done any of the following three ways

- **Buttons:** These commands will be run by all turtles
- **Commander Center:** These commands will also be run by all turtles
- **Ask turtles/hatch:** Can ask specific turtles, or create a new turtle and give it commands
Netlogo - III
When using Netlogo there are common steps that each model goes through.

1. The model is initialized with all the global variables given their initial values. Typically done with a ”setup” button.
2. Running the simulation and updating the agents is then done with a ”go” button.
Agent Sets

A set of agents of only one type of turtles/patches/links

- Use **ask** to make the agents in the agentset do something
- Use **any** to see if the agentset is empty
- Use **all** to see if every agent in an agentset satisfies a condition.
- Use **count** to find out exactly how many agents are in the set
The models in Netlogo are build on *procedures*. Each type of agent can have a procedure it calls.

For example, turtles could have procedures called *eat*, *reproduce* and *die*.

There are procedures Netlogo provides, or they can be programmed with *primitives*, which are basic commands provided by Netlogo.
Netlogo keeps track of time as discrete "ticks."

- In general, it is important to be aware of the order procedures are called.
- For example, calling die eat and reproduce is different than eat, reproduce and die.
- It is also important to be aware of what order agents are updated. If the same order is used each time, then the first agent can have an advantage over the last agent.
```
turtles-own []
  flockmates ; ; agentset of nearby turtles
  nearest-neighbor ; ; closest one of our flockmates
]

to setup
  clear-all
  crt population
    [ set color yellow - 2 + random 7 ; ; random shades look nice
    set size 1.5 ; ; easier to see
    setxy random-xcor random-ycor ]
  reset-ticks
end

to go
  ask turtles [ flock ] ; ; the following line is used to make the turtles
  ; ; animate more smoothly.
  repeat 5 [ ask turtles [ fd 0.2 ] display ]
    ; ; for greater efficiency, at the expense of smooth
    ; ; animation, substitute the following line instead:
    ; ; ask turtles [ fd 1 ]
  tick
end

to flock ; ; turtle procedure
  find-flockmates
  if any? flockmates
    [ find-nearest-neighbor
      ifelse distance nearest-neighbor < minimum-separation
      [ separate
        [ separate ]
      [ align
        cohere ]
    end
  end

to find-flockmates ; ; turtle procedure
  set flockmates other turtles in-radius vision
end

to find-nearest-neighbor ; ; turtle procedure
```
References


Netlogo webpage:
http://ccl.northwestern.edu/netlogo/