ODD protocol
Game of Thrones model Step 4

OVERVIEW

Purpose

This model explores the demographic dynamics between sheep, wolves, and humans. This is the fourth step towards creating a model of Westeros. It is based on the Wolf Sheep Predation model (Wilenski 1997). In this new step, the model now includes cities and roads connecting them. The major cities that are connected by major roads are also forming a network (which can be either directed or undirected) that can be analyzed using different centrality measures.

Entities, state variables, and scales

This model uses the wolves and sheep agents of the Wolf Sheep Predation model. To those, it adds a breed of human agents who can feed on grass or sheep, and who domesticate some of the wolves they encounter. It also adds a breed of cities that are sworn to a certain house and have a certain size.

Table 1. Global variables used in this model.

Global variables	**Explanation**
Max-sheep	To stop the simulation if there are too many sheep
Land-patches	Identifies that patches that are land (can be walked on)
Initial-number-sheep	Determines the number of sheep at setup
Initial-number-wolves	Determines the number of wolves at setup
Initial-number-humans	Determines the number of humans at setup
Grass-regrowth-time	After how many ticks grass regrows (resource)
Sheep-gain-from-food	How much energy sheep get from grass
Wolf-gain-from-food	How much energy wolves get from eating a sheep
Rate-sheep-eating	How often humans eat an encountered sheep
Sheep-reproduce	Each sheep's probabilities of reproducing if they have energy
Wolf-reproduce	Each wolf's probabilities of reproducing if they have energy
Humans-reproduce	Each human's probabilities of reproducing if they have energy
Show-energy?	Switch, determines if we see sheep's energy on the window
Link-type	Determines the type of edges created between cities (directed or not)
Centrality-type	Determines which centrality measure is calculated between cities

Table 2. Turtle variables

turtle variables	**Explanation**
Energy	Counter that gets updated when an agent moves and eats. At 0, the agent dies.
Wolves variables	
Domesticated?	True/False. Record if a wolf has been domesticated.
Cities variables	

City-name	The name of the city or castle (from GIS data)
City-size	The size of the city or castle (from GIS data)
Centrality	Takes on the centrality measure calculated (only if part of network)

Table 3. Patch variables

Patch variables	**Explanation**
countdown	Counter used to regrow grass.
Land?	Boolean that identifies if the patch is land
Wall?	Boolean that identifies if the patch overlaps with the wall (and thus becomes it)
House	Identifies which house owns the patch
House-color	Records the color linked to the house (for visibility purposes)
Road?	Identifies if the patch is part of a road

Process overview and scheduling

This model is pretty simple and is the same as in Step 3. At every tick, all agents move, which depletes their energy, and they look for food. If their energy is depleted, they die. If they still have energy, they consider reproducing.

Ticks do not represent any specific unit of time.

DESIGN CONCEPT

Basic principles

This model added cities and roads to the previous model, which had added humans to the Wolf Sheep predation model that aimed to show the predator-prey equilibrium that can arise between two species.

Emergence
None.

Adaptation

None.

Objectives

As in previous versions, humans, sheep, and wolves have the objective to eat and reproduce. Cities do not have objectives yet. In general, this version of the model simply adds connected cities and is done to help the user explore how to create and analyze networks in NetLogo.

Learning

None.

Prediction

None.

Sensing

Agents can sense if they have reached water or the edge of the map, in which case, they turn around. Cities can sense who they are connected to via links.

Interaction

Wolves and humans interact with sheep by eating them. Humans interact with wolves by either domesticating them or getting eaten by them.

Stochasticity

Only the energy level of the agents is set randomly at the beginning of a simulation (with a maximum value set by a slider). Which patches are depleted at setup is also random, as well as the regrowth countdown of depleted patches.

Collectives

Land-patches identifies the set of patches that can be walked through and that have resources to deplete.

Observation

There is one plot that follows the population sizes of sheep, wolves, dogs, and humans.

DETAILS

Initialization

At setup, the model imports a raster and resizes the World window to fit the new map's dimensions. The raster's values represent the house-color of the seven kingdoms. All patches record the raster's value as their house-color. Patches with house-color above 0 are land patches, whereas others are water. Water patches set their colors to blue. The model then imports a line vector that covers the extent of the wall. All patches that intersect that line become the wall. They set their 'land?' variable to *false* to make sure that they cannot get walked through. Wall patches are white. The model then imports a political map, which holds

vector polygons of the different kingdoms with specific information. For the moment, patches that intersect with any polygon record that polygon's 'claimedBy' value, which is the identity of the house that owns the land (e.g. Stark, Tyrell,…). All water patches are claimed by 'no one'. As there is a bit of disconnect between the polygon vector and the Westeros raster, the model tells any patch who thinks it is water but that overlaps with a house polygon to simply forget about that house and instead be claimed by 'no one'. The model then imports a vector file that contains the locations of cities and castles and creates cities there. Cities' sizes are twice the value of the vector's 'size' variable, whereas their color is based on the 'house-color' of the patch on which they are created. Finally, the model imports another vector files, which has lines representing the major roads. All patches intersecting with those lines set their 'road?' variable to *true*, set their color to black and do not get resources.

About half the land patches that are not wall or road become green, which means that they have resources. The rest get a countdown value that determines when their grass will grow back. The number of sheep, wolves, and humans are determined by sliders. They are placed randomly on the land, with a random energy value with max based on a slider. Humans are colored based on the territory they were 'born' into (e.g., Starks are grey).

Input data

This model uses five GIS maps. These maps were taken from https://www.cartographersguild.com/showthread.php?t=30472 and modified to suit our modeling needs.

- Land_raster.asc: raster map with the values representing the color number (in NetLogo swatch) of the house that owns the land.
- Wall.shp: line vector that covers the extent of the wall.
- W_political_revised.shp: Polygon vector. Each polygon represents a house. It holds the following information for each polygon:
    - Name: Name of the kindgom (e.g., "The North","Crownsland",…)
    - Population: Estimated millions of people living in each kingdom (based on https://atlasoficeandfireblog.wordpress.com/2016/03/06/the-population-of-the-seven-kingdoms/)
    - House-color: NetLogo number associated with each house's colors (arbitrary).
    - claimedBy: The name of the house that claims each kingdom (at the beginning of A Song of Ice and Fire).
- Westeros_locations.shp: Point vector of important locations. It holds the following information for each point:
    - Type: The type of location (e.g., Castle, city, ruin, other, …).
    - Name: Name of the location (e.g., Castle Black, Winterfell, …).
    - Size: An assigned size (scale of 1-5) for each location. We assume it represent the population size of each.
- Westeros_roads.shp: Line vector of major roads. While it holds a few variables (name and size), we do not use them in our model.

Submodels

Move:
This is called by all turtles. The turtles wiggle a bit. If the patch ahead is either the edge of the world or water, they turn 180 degree. They then move forward by 1.

Eat-grass:
This is called by sheep. The patch becomes brown and the sheep adds a certain value (based on slider) to its energy level.

Eat-sheep:
This is called by wolves. If there is a sheep on the same patch as the wolf, the sheep dies and the wolf adds a certain value (based on slider to its energy level).

Death:
Called by all agents. If energy reaches 0, the agent dies.

Grow-grass:
Called by all land patches. If the patch is brown, it checks its counter. If the counter is at 0, the patch becomes green. If not, the counter is updated to its value -1.

Display-labels:
Called only if the switch is ON. It shows the energy level of wolves and sheep.

Add-humans:
Called during setup. Creates a certain number of humans. They set their colors based on the house that owns the patch where they were born.

Reproduce:
Called by all agents. They roll a die. If it is successful, they create a new agent and give them half of their energy.

Humans-eat:
Called by humans. A certain percentage of the time, they will look for a sheep to eat. If there is one on the same patch, that sheep dies, and the human adds a certain value (based on the wolf-gain-from-food slider) to its energy level.

When the human does not look for a sheep, it looks at the grass. If there is grass on the patch it is on, the patch becomes brown and the human adds a certain value (a quarter of what sheep gain from grass, as humans cannot convert plants to energy as well as animals) to its energy level.

Domesticate-wolves:

Called by humans. If there is a wolf on the same patch, and that wolf is still wild, the human rolls the dice. About 50% of the time, it will domesticate the wolf. The remaining times, it will get eaten by the wolf (who gets 1.5 times energy as it would from a sheep, arbitrary decision).

Humans-feed-animals:
If there is a domesticated wolf on the same patch (aka, a dog), it will ask the dog to feed on a sheep if there is one around, but the human will take some of that energy (they share the sheep).

Calculate-centrality:
This is called by an Interface button. It asks all the cities that have a link to calculate their centrality and change their size to reflect said centrality. The type of centrality varies based on the ***centrality-type*** chooser. It can be either degree, closeness, betweenness, or eigenvector.

REFERENCES CITED

Wilensky, U. (1997). NetLogo Wolf Sheep Predation model. http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Most of the ideas modeled here come from the A Song of Ice and Fire books, written by George R.R. Martin.

Some ideas and concepts (colors of houses, for example) are inspired by the HBO Game of Thrones show.