



EPIc Series in Computing

Volume 58, 2019, Pages 257–264

Proceedings of 34th International Conference on Computers and Their Applications



NetLogo and GIS: A Powerful Combination

Broday Walker and Tina V. Johnson

Department of Computer Science, Midwestern State University
Wichita Falls, TX 76308, USA
(bkwalker0728, tina.johnson)@msutexas.edu

Abstract

NetLogo is a popular agent-based modeling system for good reason. It is relatively easy to learn; it allows an intuitive user interface to be built with predefined objects, such as buttons, sliders, and monitors; and available documentation is extensive, both on the NetLogo Website and in public forums. The Geographic Information Systems (GIS) extension for NetLogo allows real-world geographic or demographic data to be incorporated into NetLogo projects. When GIS is combined with NetLogo, simulations can be transformed from a basic representation to one that accurately replicates the characteristics of a map or population. This paper describes the necessary steps for incorporating GIS within a NetLogo project and the primitive commands used for associating shape properties to NetLogo patches. A practical example is included that demonstrates how to import a map of Texas into a NetLogo project and use the vector data in conjunction with NetLogo patches to randomly color each county.

keywords: Agent-based modeling, ABS, NetLogo, GIS

1 Introduction

Agent-based modeling (ABM) is particularly useful for simulating complex systems in which agents interact with each other within a given environment. The collective behavior of individual agents gives rise to patterns which might otherwise go unrecognized. Although ABMs have been criticized in the past for a lack of reproducibility, recent models have become more formalized and recognized as a scientific basis for exploring complex systems [2].

NetLogo, developed by Uri Wilensky at Northwestern University's Center for Connected Learning and Computer-Based Modeling, is an expert tool for ABM [7, 10]. It has been used in a variety of disciplines, including biology, environmental modeling, neural computing, and many others [4, 5, 1]. One of the most powerful aspects of NetLogo is the ability to extend its capabilities by integrating with other languages, such as Python and R [3, 6]. This paper provides a brief introduction to NetLogo and the Geographic Information Systems (GIS) extension, which allows vector and raster data to be directly incorporated into the NetLogo environment.

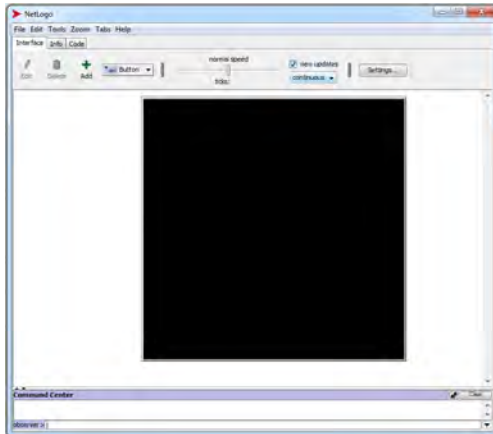


Figure 1: An empty NetLogo project

```

to setup
  clear-all
  create-turtles 50 [setxy random-xcor random-ycor]
end

```

Figure 2: Generic NetLogo procedure associated with a setup button

2 NetLogo Basics

NetLogo has an intuitive user interface that consists of a main menu and three tabs: Interface, Info, and Code (see Figure 1). The Interface tab (currently active in Figure 1) is used to create a Graphical User Interface (GUI) for the model being developed. The Info tab is for project documentation and the Code tab is for the actual model code.

Several GUI objects are available from a dropdown menu in the Interface tab, including buttons, sliders, monitors, plots, and more. Objects placed in the user interface are connected to code. For example, a button on the interface is associated with a procedure in code; sliders and monitors are associated with variables, and plots use variables to render graphs during execution. This direct connection between the user interface and code allows a sophisticated interface to be readily developed.

Agents in NetLogo, referred to as turtles, move in a world of patches. Both turtles and patches can be programmed to meet the requirements of the model. Turtles may represent any type of agent. For example, in the Wolf Sheep Predation simulation [9] (one of over 100 models that are downloaded with NetLogo), turtles represent both wolves and sheep, and patches represent land which may or may not contain grass. Patches of grass may be eaten by sheep and sheep may be eaten wolves. In the Wave Machine simulation [8], agents represent components of a membrane and patches make up a frame holding the edges of the membrane.

Procedures in NetLogo form the basic building blocks for writing code. Procedures can be called by clicking on a button or by an explicit procedure call. A procedure begins with the keyword *to*, followed by the name of the procedure, the commands that make up the body of the procedure, and the keyword *end*. If a procedure is to be called by the click of a button in the interface, the button must stipulate which procedure to invoke. For example, many models have a *setup* button that is used to initialize the simulation. Figure 2 demonstrates a generic setup procedure that clears the world (the black box shown in Figure 1) and creates 50 turtles which are placed in random positions in the environment.

In addition to an extensive Models Library, which contains modifiable, fully-functioning models, the NetLogo website contains a comprehensive dictionary and step-by-step tutorials. Additionally, there are numerous NetLogo resources through forums and publications. Programmers who wish to learn NetLogo have an abundance of options from which to choose.

3 GIS Extension

The GIS extension for NetLogo provides users with the ability to import GIS vector and raster data to be used in conjunction with the myriad standard features already present in NetLogo. A typical NetLogo simulation utilizes turtles and square patches that interact using predefined rules. Turtles and patches are aware of each other and able to interact and change as conditions/variables change. These agents may ask other agents to perform a task or procedure. It is possible to allow the world to wrap in a torus shape and to modify turtle shapes, but the shape of the patches is fixed as a square [10]. While these characteristics are sufficient for many tasks, the GIS extension is a powerful tool for creating agent-based models that are extended to include the use of accurate maps as well as real-world datasets. The extension creates the ability for vector data to interface with agents in NetLogo much like patches and turtles already interact. This article focuses on the basic procedures required to import and manipulate a vector dataset inside a NetLogo model using the GIS extension.

Getting started with the GIS extension requires creating or finding a map and its associated data. There are several options available for accomplishing this task, including ArcGIS, which is a paid service, and qGIS, which is open source and free to use. Using a GIS application allows the user to modify maps found online to include only the information needed to run their NetLogo model. These applications also allow the user to make their map compatible with the NetLogo space. Some maps must be converted to a supported extension. Currently, the GIS extension supports the use of ESRI shapefiles (.shp) for vector files and ESRI ASCII Grid files (.asc and .grd) for raster data as documented under the GIS Extension link found on the NetLogo website [10].

Free and open source maps are available from numerous government agencies, including the United States Geological Survey and the United States Census Bureau. These two resources are invaluable when starting with GIS in NetLogo, as finding a quality vector or raster map to use is not always a trivial task. The maps and data given by the U.S. Census bureau are also valuable as they contain a wealth of real-world data about the United States population, which can be useful in making agent-based models that are incredibly detailed and realistic. In this way, the GIS extension offers the opportunity to explore problems and solutions within a realistic and representational framework.

4 Texas Counties Example

Once the desired map has been selected and the appropriate modifications have been made, the map can be imported into a NetLogo model. The Texas counties model is included to illustrate the usefulness of GIS. This model is a practical example that demonstrates tasks such as importing datasets, assigning vector features to individual patches, drawing polygons, and filling polygons with different colors based on the value of a variable. Interfacing between the NetLogo model patches and the vector files is also discussed.

The GIS extension is bundled with the standard NetLogo software download and can be introduced into the project by including *extensions [gis]* at the beginning of the code (see Figure 3). This directive is included at the top of the code much like including a library in any program.

Once the GIS commands are available, it is necessary to define one or more global variables that will hold the imported data. As shown in the code, this declaration follows after the inclusion of the GIS extension. NetLogo requires global variables to be declared at the beginning

```

extensions [ gis ]

globals [ tx-counties ]

patches-own [
  random-n
  centroid
  ID
]

```

Figure 3: NetLogo extensions and variables

```

to setup-map
  ;; Load the dataset
  set tx-counties gis:load-dataset "data/tx-counties.shp"
  ;; Set the world envelope
  gis:set-world-envelope (gis:envelope-of tx-counties)
end

```

Figure 4: setup-map procedure

of the program before any functions are defined. These variables are able to be accessed and used by every patch and agent in the simulation [10]. In the Texas counties model, the global variable being used is named *tx-counties* (see Figure 3). More than one dataset may be included in a model by using different global variables to hold their data.

Ideally, the map setup procedure is isolated from the setup procedures required for the model. In other words, it may be helpful to include a setup-maps button and a separate setup-model button. Because the map, its lines, polygons, and associated data will stay the same from run to run, it only needs to be imported and assigned to its respective global variable one time before the first model run. Executing this procedure once saves computation time as well as time spent drawing each county, as this is much more time consuming than clearing turtles or patches. This portion of the model is contained in the *setup-maps* procedure. This procedure loads the dataset, assigns it to a global variable, then loops through the imported vector feature list and assigns each one to the patch or patches intersecting that feature.

To accomplish the tasks set out in the *setup-maps* procedure, some GIS extension primitives must be used. A primitive performs a predefined function specific to the GIS extension. With these, the user can query vector features or entire datasets to determine if one polygon contains another, if a patch intersects a specific vector feature, or a limited number of other GIS operations [10]. GIS primitives are accessed by using the prefix *gis* followed by a colon and the primitive to be used. To use the data from the vector or raster dataset, the *gis:load-dataset* command is used to assign the dataset to the previously defined global variable named *tx-counties*. As shown in Figure 4, the full command is *set tx-counties gis:load-dataset "data/tx-counties.shp"* (see figure 4). The included file path specifies the location of the vector or raster file to be used and any associated projection and database files. If the files are located in the same directory as the NetLogo model, it is not necessary to include the entire file path.

Once the vector dataset has been assigned to a global variable, the user must set the world envelope. According to the GIS Extension documentation on the NetLogo website, “The easiest way to define a transformation between GIS space and NetLogo space is to take the union of the “envelopes” or bounding rectangles of all of your datasets in GIS space and map that directly to the bounds of the NetLogo world [10].” Setting the world envelope is necessary to ensure the vector data is properly associated with the model’s world. To set the world envelope, the *gis:set-world-envelope* command is used. It is possible to nest this command multiple times to include all datasets being used. The *gis:set-world-envelope* command defines a transformation between the NetLogo space and the GIS data space being used in the model. The Texas Counties model only requires one transformation to be completed (see figure 4).

After the dataset has been loaded and the world is ready for use, a foreach loop is used to assign data to the patches that intersect the vector features. Depending on the data used, a feature may be a polygon or a line that represents anything from a road, to a river, and even

a state. The Texas counties model encompasses the state of Texas and its 254 counties, each represented by one polygon. Before the foreach loop begins, a local counter variable is defined and initialized to the value of 1. This variable will be incremented by 1 at the end of each cycle through the loop. It will also be used to set the ID of the patches intersecting a certain feature.

The foreach loop begins by utilizing the *gis:feature-list-of tx-counties* command (see Figure 5). As the name implies, the *tx-counties* global variable holds a list of all the features contained in that dataset. Because this model has very little data associated with it, the feature list is the collection of all the polygons that make up the Texas counties map. These features will be passed to the anonymous procedure inside the loop, which will accomplish the task of taking data from the *tx-counties* dataset and assigning it to the underlying patches.

Once inside the loop, the anonymous procedure operator is used. Anonymous procedures are a valuable resource when used in conjunction with the GIS extension because they provide a simple avenue for looping through a list and doing a certain task specified for (or with) each item in the list. The basic structure for an anonymous procedure in NetLogo follows the following format: `[[argument(s)] -> commands]` [10].

Further into the foreach loop, the patches intersecting the current feature in the list are asked to set their variable named centroid to the location of the centroid of the polygon that overlays them. Two GIS primitives, *gis:location-of* and *gis:centroid-of* are used to find the centroid of the polygon. The patch at the centroid of the polygon will serve as a command center for that polygon. While this seems counter-intuitive at first, it is a necessary step to take as it is not currently possible to ask a polygon to run a procedure or complete a set of tasks in the same way that patches, turtles, and agentsets can be addressed. This patch will set its ID variable to the value of *i* as well as inherit any variable values assigned to it in the future. For example, this patch could have a variable that is assigned a number that represents the population of the county. Other patches under the polygon will not receive this value because the centroid patch is being used to control the polygon. These ID patches are also used when calling upon GIS drawing primitives, greatly easing the process of redrawing the filler color of each polygon at the end of each tick.

After the loop finishes, the Texas county lines are drawn using the *gis:set-drawing-color* and *gis:draw* commands (see Figure 6). This is a simple process that includes selecting the color to be used and the dataset to draw. It is not necessary to loop through each feature in the list of features for the dataset to be drawn. Rather, it is only necessary to specify which dataset to draw and the width of the line to be used. In the Texas counties example, the county lines are drawn in white with a width of one pixel.

It is important to draw attention to a potential problem that can be quite difficult to spot when assigning identification numbers to a patch underlying the centroid of a polygon. There is a chance in NetLogo, which uses the value of 16 as the maximum x- and y-coordinates by default, for multiple polygons to possess a centroid value that falls on the same patch. When this issue occurs, it is impossible to determine which values will be assigned from the conflicting polygons to the underlying patch. This is typically seen when trying to draw or fill in the polygons that are in conflict with each other. When this issue occurs, it is common to see black, unfilled polygons or a group of polygons assigned with the incorrect color. The Texas counties model suffers from this complication at the default NetLogo world size. To remedy this problem, the model was modified and enlarged to have maximum x- and y-coordinates of 24 units, resolving the centroid conflicts between polygons. To change the size of the NetLogo world space, simply navigate to the Interface tab, which is home to the model world, and select the Settings button, as shown in Figure 7. Once the changes are made, the model is set up and ready to be used to run simulations. The previous set of tasks only need to be completed one time. Now that map

```

;; Loop through the patches and find centroid and set ID
let i 1
foreach gis:feature-list-of tx-counties [ feature ->
  ask patches gis:intersecting feature [
    set centroid gis:location-of gis:center-of feature
    ask patch item 0 centroid item 1 centroid [
      set ID i
    ]
  ]
  set i i + 1
]

```

Figure 5: Procedure to associate patches to county centroids

```

;; Draw the outline of the counties in white
gis:set-drawing-color white
gis:draw tx-counties 1
end

```

Figure 6: Commands to draw county lines

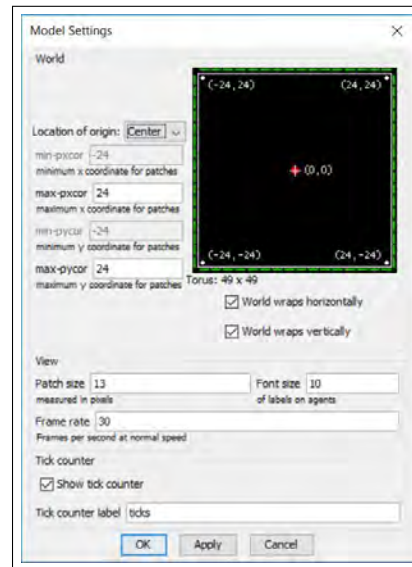


Figure 7: Dialog box for changing world size

setup is complete, the rest of the model can be setup.

With the model world properly setup, the setup for each model run must be completed. Each county in the map of Texas contains a variable at the centroid patch which is assigned a new random number between 0 and 10 during each tick. In this toy model, the random number is used to control the changing of each filler color for the counties. If the number is less than 5, the centroid patch instructs the polygon to fill itself with the color blue. If the number is greater than or equal to 5, the polygon will fill itself with the color red. While this is a toy example, it demonstrates the ability of the GIS extension to be used for dynamically changing the color of the space contained within a certain polygon based on ever-changing information.

Re-drawing the filler color for each polygon can be accomplished with basic logic as shown in Figure 8. In this case, the syntax changes slightly when addressing the individual polygons rather than the entire list of features. Each centroid patch is asked to update the polygon it controls to the appropriate color. The color is selected by using the *gis:set-drawing-color* primitive. To fill the polygon, the *gis:fill* primitive is used along with the item specified. To control this portion of the color-change procedure, the ID variable is used. Because each polygon is contained in the feature list of the *tx-counties* dataset, the ID must be subtracted by 1 when accessing it in this procedure due to NetLogo lists beginning at element 0. For example, the polygon that has a centroid patch with ID = 1 is the 0th element in the feature list of the *tx-counties* dataset. The polygon is then filled with the appropriate color with a line thickness of 2.0 pixels.

At this stage, the model is complete and ready to run. Using the random numbers that are assigned to each centroid patch every tick, the counties will update their color and change according to their assigned value. Though the random numbers are not associated with real data in this demonstration, the key link for communication between imported vector data and the NetLogo world has been established and can be extended to other models. Additionally, data generated from the simulation can be exported for use in GIS software. Figure 9 shows a

```

to color-change
  ask patches with [ID > 0] [
    set random-n random-float 10
    ifelse random-n >= 5
    [
      gis:set-drawing-color red
    ]
    [
      gis:set-drawing-color blue
    ]
    gis:fill item (ID - 1)
    gis:feature-list-of tx-counties 2.0
  ]
end

```

Figure 8: Procedure to change county colors

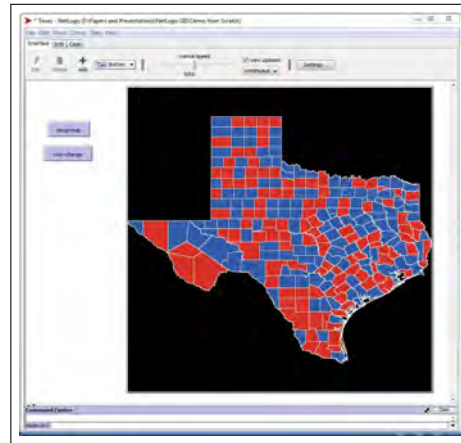


Figure 9: NetLogo-GIS integration example

typical output for this simple NetLogo project.

5 Conclusion

NetLogo is a powerful ABM simulation tool. It allows a programmer to design a visual model of interacting agents in a simulated environment. Additional libraries can be incorporated to extend the capabilities of the basic NetLogo model. The GIS extension discussed in this paper is useful for providing a realistic environment within the NetLogo world. Although the example in this paper used a shape file of a Texas map, any NetLogo supported vector format may be incorporated. Pairing NetLogo with GIS greatly increases the look and functionality of location-based simulations. The complete code for the project and the associated files referred to in this paper can be found at .

References

- [1] Thomas Banitz, Anna Gras, and Marta Ginovart. Individual-based modeling of soil organic matter in netlogo. *Environ. Model. Softw.*, 71(C):39–45, September 2015.
- [2] Volker Grimm, Uta Berger, Donald Deangelis, J Polhill, Jarl Giske, and Steven F. Railsback. The odd protocol: A review and first update. *Ecological Modelling*, 221:2760–2768, 11 2010.
- [3] Marc Jaxa-Rozen and Jan H. Kwakkel. Pynetlogo: Linking netlogo with python. *J. Artificial Societies and Social Simulation*, 21(2), 2018.
- [4] Cristian Jimenez-Romero and Jeffrey Johnson. Spikinglab: modelling agents controlled by spiking neural networks in netlogo. *Neural Computing and Applications*, 28(1):755–764, Dec 2017.
- [5] Jerry Rhee and Philip Iannaccone. Mapping mouse hemangioblast maturation from headfold stages. *Developmental biology*, 365:1–13, 02 2012.
- [6] Jan C. Thiele. R Marries NetLogo: Introduction to the RNetLogo Package. *Journal of Statistical Software*, 58(2), 2014.
- [7] Seth Tisue and Uri Wilensky. Netlogo: A simple environment for modeling complexity. In *International Conference on Complex Systems*, pages 16–21, 2004.

- [8] U. Wilensky. Netlogo wave machine model. <http://ccl.northwestern.edu/netlogo/>, Center for Connected Learning and Computer-Based Modeling, 1997.
- [9] U. Wilensky and K. Reisman. Connected science: Learning biology through constructing and testing computational theories - an embodied modeling approach. *International Journal of Complex Systems*, 234:1–12, 1999.
- [10] Uri Wilensky. Netlogo. <http://ccl.northwestern.edu/netlogo/>, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL, 1999.